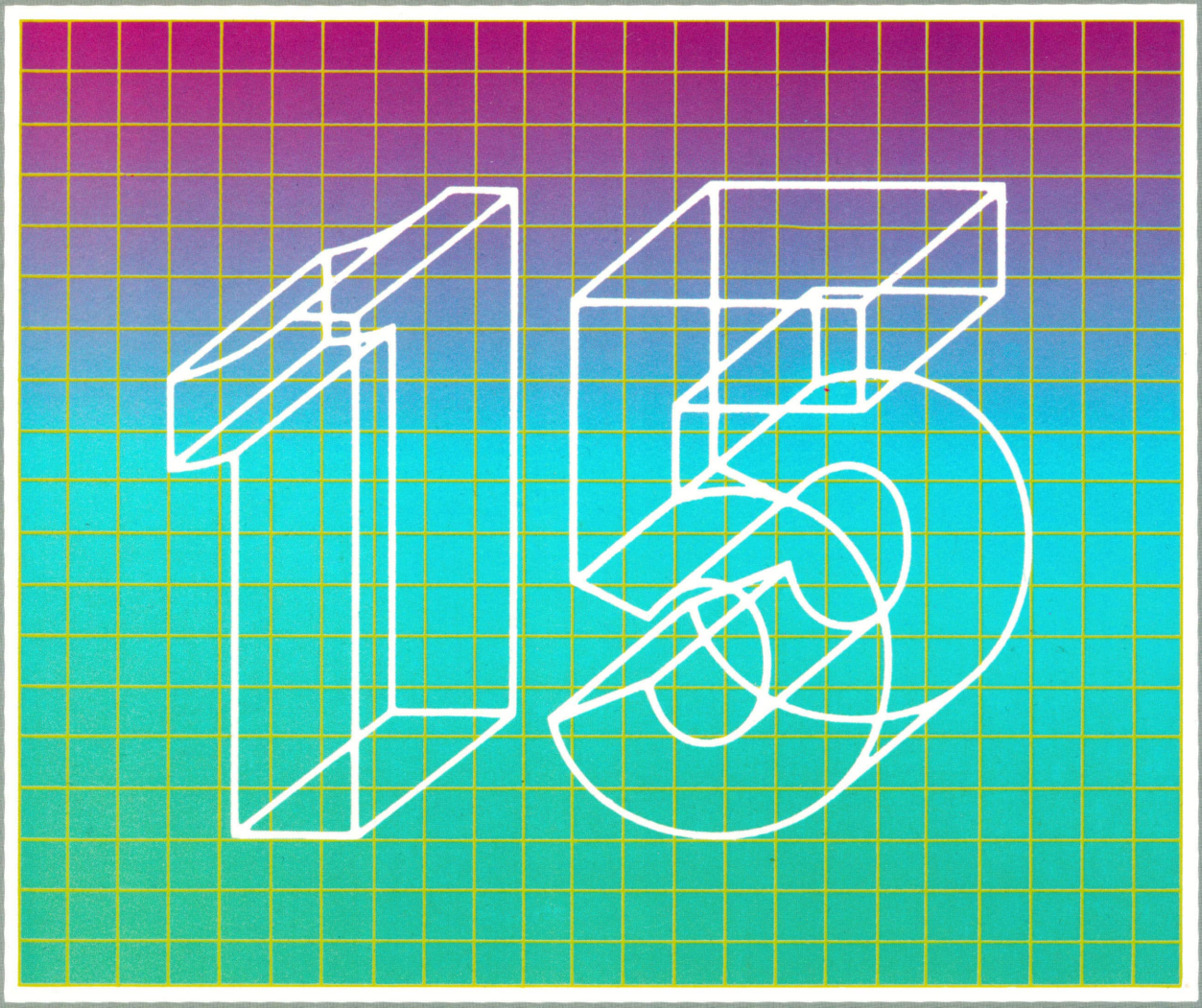
BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD, IBM, SPECTRUM, COMMODORE Y MSX



BRICOLAGE INFORMATICO

MANEJO DE ELEMENTOS GRAFICOS

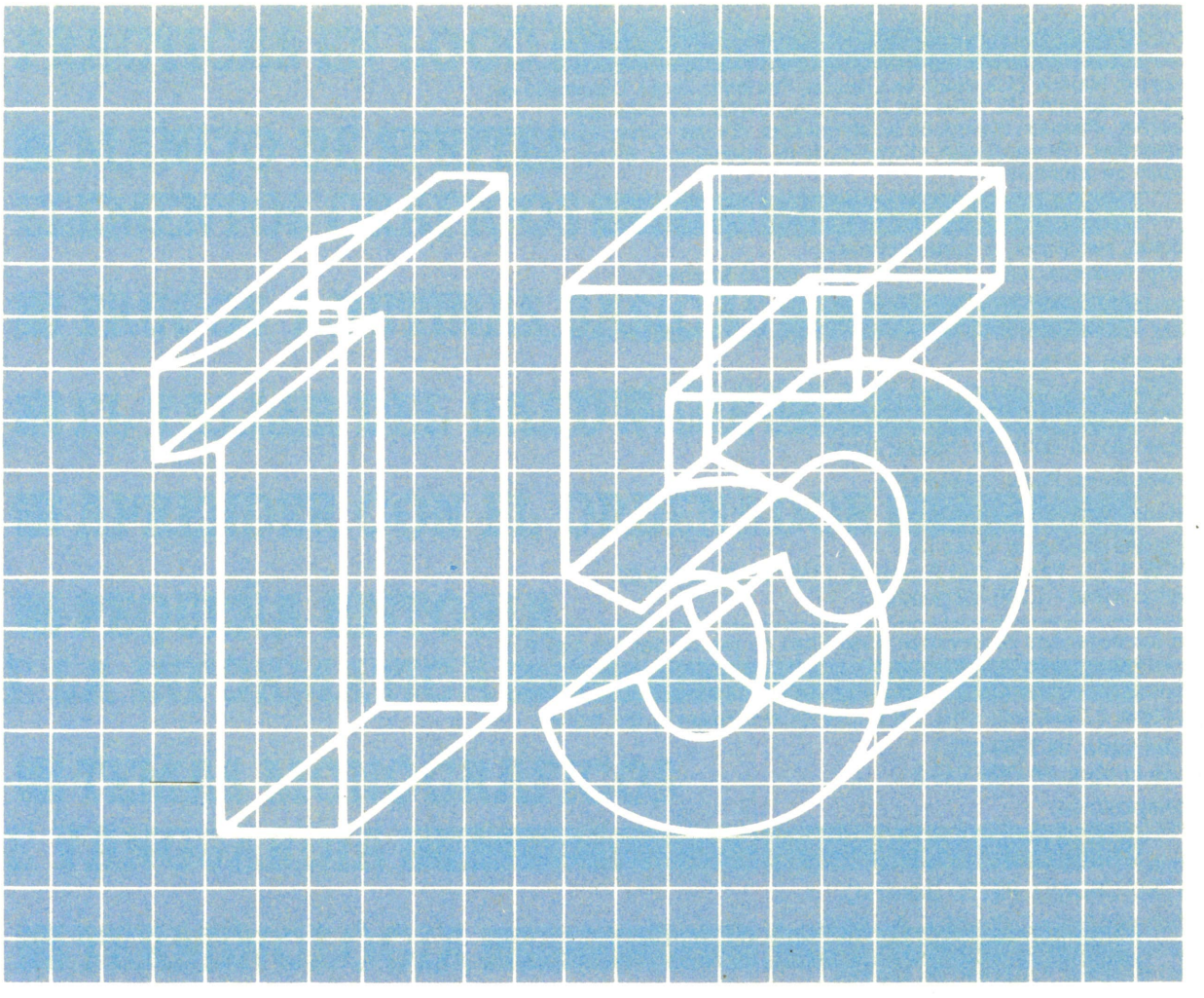
EDICIONES SIGLO CULTURAL

BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD, IBM, SPECTRUM, COMMODORE Y MSX



EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S. A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Director de la colección:
JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño:
BRAVO-LOFISH.

Maquetación:
D. S. M.

Dibujos:
JOSE OCHOA

Tomo 15. «Experiencias y prácticas en logo», Carlos Albert, Técnico de Informática; Adoración Llena Jubero, Técnico de Informática. «Manejo de sprites y elementos gráficos», «Trucos y rutinas básicas», Francisco Morales. Técnico de Informática. «Aprende con el ordenador», AULA DE INFORMATICA APLICADA: Soledad Tamariz-Martel, Diplomada en Telecomunicación; Francisco Blanca, Diplomado en Telecomunicación; Alejandro Marcos, Licenciado en Química; Fernando Suero, Diplomado en Telecomunicación. «El Taller del Hardware», Carlos Rey, Ingeniero Industrial. «Pequeña historia de la Informática», «Temas monográficos de vanguardia», «Terminología», «Vocabulario de Informática», Blanca Arbizu, Técnico de Informática.

Ediciones Siglo Cultural, S. A.

Dirección, redacción y administración:
Pedro Teixeira, 8-2.°. Teléf. 810 52 13. 28020 Madrid.

Publicidad:
Gofar Publicidad, S. A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S. A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

Todos los derechos reservados. Este libro no puede ser, en parte o totalmente, reproducido, memorizado en sistemas de archivo, o transmitido en cualquier forma o medio, electrónico, mecánico, fotocopia o cualquier otro, sin la previa autorización del editor.

ISBN del tomo: 84-7688-087-1
ISBN de la obra: 84-7688-047-2

Fotocomposición:
ARTECOMP, S. A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. PINTO (Madrid).

© Ediciones Siglo Cultural, S. A., 1986
Depósito legal: M-43.185-1986
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S. A.
Pedro Teixeira, 8-2.°. Teléf. 810 52 13. 28020 Madrid.

Mayo, 1987

P.V.P. Canarias: 355.—Ptas.

INDICE

EXPERIENCIA Y PRACTICAS EN LOGO 5

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS 15

TRUCOS Y RUTINAS BASICAS 23

EL TALLER DEL HARDWARE 29

APRENDER CON EL ORDENADOR 37

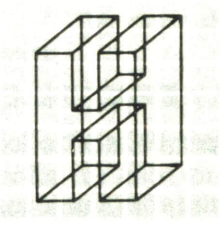
PEQUEÑA HISTORIA DE LA INFORMATICA 49

TEMAS MONOGRAFICOS DE VANGUARDIA 53

TERMINOLOGIA 56

VOCABULARIO DE INFORMATICA 59

Incrementando el valor de una variable



ASTA ahora cuando realizábamos ejercicios en los que incrementábamos el valor de una variable en 1, lo hacíamos a través de otra variable, que podríamos llamar auxiliar por la función que desempeña, veamos ahora una forma mucho más sencilla.

Asignamos a la variable A el valor 0.

```
?HAZ "A 0
?ESCRIBE :A
0
```

A la variable A le sumamos 1 y el resultado de la suma lo guardamos en A.

```
?HAZ "A :A + 1
?ESCRIBE :A
1
```

El orden que sigue el ordenador para realizar la orden HAZ "A :A + 1, es el siguiente:

— Calcula el valor de la expresión :A + 1; para ello irá a buscar el valor que tiene la variable A y sumará a dicho valor el valor 1.

— Almacena el resultado de la expresión en A.

Incrementemos tres veces el valor de una variable en uno.

```
?HAZ "A 0
?ESCRIBE :A
0
?HAZ "A :A + 1
?ESCRIBE :A
1
?HAZ "A :A + 1
?ESCRIBE :A
2

?HAZ "A :A + 1
?ESCRIBE :A
3
```

Otra forma de hacerlo sería:

```
?HAZ "A 0
?ESCRIBE :A
0

?REPITE 3 [HAZ "A :A + 1 ESCRIBE :A]
1
2
3
```

Observa que siempre inicializamos la variable A, es decir, le asignamos un valor inicial.

```
?HAZ "A 30
?REPITE 10[HAZ "A :A + 1 ESCRIBE :A]
```

obtenemos los diez números naturales que siguen al 30.

Ahora vamos a obtener la tabla de multiplicar de cualquier número.

Veamos, primero, cómo sería el procedimiento para obtener la tabla del cuatro.

```
?PARA TABLA4
>HAZ "C 0
>REPITE 10 [HAZ "C :C + 1 ESCRIBE
```

Es sencillo incrementar el valor de una variable.

```
BE: C * 4]
>FIN
```

Compruébelo tecleando TABLA 4.

```
?TABLA4
4
8
12
16
20
24
28
32
36
40
```

Y la tabla de cualquier número N.

```
?PARA TABLA:N
>HAZ "C 0
>REPITE 10 [HAZ "C :C + 1 ESCRIBE:
C * N]
>FIN
```

Que queremos la tabla del cinco.

```
?TABLA 5
```

la del 9

```
?TABLA 9
```

si las queremos todas, definimos un procedimiento para obtenerlas.

```
?PARA TABLAS
>HAZ "T 0
>REPITE 10 [HAZ "T :T + 1 TABLA :T
ESPERA 30]
>FIN
```

y lo ejecutamos

```
?TABLAS
```

Seguro que ya te has hecho una idea de la utilidad de este tipo de expresiones.

También podemos decrementar el valor de una variable con el mismo proceso.

```
?HAZ "A 30
?HAZ "A :A - 1
?ESCRIBE :A
29
?HAZ "A 10
?ESCRIBE :A
?REPITE 10 [HAZ "A :A - 1 ESCRIBE :A]
```

hemos obtenido los números naturales del 10 al 0.

En el lenguaje coloquial informático, estas variables reciben el nombre de variables contadoras, no porque la variable en sí sea especial, sino porque este tipo de juego que se realiza con la misma variable se utiliza esencialmente en aquellos procesos que requieren contabilizar algo.

Se puede contar de uno en uno como ya hemos visto y se puede contar de dos en dos, de tres en tres,...

```
?HAZ "B 0
?REPITE 100 [HAZ "B :B + Z ESCRIBE :B]
```

Así de sencillo es obtener la lista de los 100 primeros números pares.

Para obtener los 100 primeros números impares lo único que variaremos será la inicialización de la variable B.

```
?HAZ "B 1
?REPITE 100 [HAZ "B :B + 2 ESCRIBE :B]
```

¿Y para obtener números decimales?

```
?HAZ "C 0
?REPITE 10 [HAZ "C :C + 0.1 ESCRIBE
:C]
```

Con estas dos líneas conmseguimos 10 números del 0 al 1.

Veamos cómo podemos obtener los cuadrados de los 10 primeros números naturales.

```
?PARA CUADRADOS
>PT
>ESCRIBE "CUADRADOS
>HAZ "C 0
>REPITE 10 [HAZ "C (:C + 1) HAZ "C2
(:C * :C) ESCRIBE :C2]
>FIN
```

Los ejecutamos:

```
?CUADRADOS
```

Este tipo de operaciones que estamos realizando con una misma variable, simplifica

También podemos decrementar el valor de una variable.

los cálculos en muchos procesos informáticos.

Imagina un proceso en el que necesitemos obtener la suma de los N (estamos indicando un procedimiento con una entrada N), primeros números naturales.

```
PARA SUMANUM :N
>HAZ "C 0
>HAZ "SUMA 0
>REPITE :N [HAZ "C :C + 1 HAZ "SUMA
:SUMA + :C ESCRIBE :SUMA]
>FIN
```

Como ya sabíamos obtener los N primeros números naturales, lo único nuevo que añadimos a este procedimiento es la acumulación en la variable (SUMA) de los números naturales.

La suma de los 40 primeros la obtendremos:

```
?SUMANUM 40
```

Un ejemplo donde utilizaríamos la filosofía de este procedimiento es en el cálculo del factorial de un número. Unicamente variaremos al orden (HAZ "SUMA :SUMA + :C) por la orden (HAZ "SUMA :SUMA * C),la inicialización de la variable SUMA y que sólo escribimos en el resultado final.

```
?PARA FACTORIAL :N
>HAZ "C 0
>HAZ "SUMA 1
>REPITE :N [HAZ "C :C + 1 HAZ "SUMA
:SUMA * :C]
>ESCRIBE :SUMA
>FIN
```

El factorial de 10 es:

```
?FACTORIAL 10
```

Cuando vimos las funciones matemáticas dijimos que no todas las versiones del LOGO disponen de todas ellas; si la versión que estás utilizando no dispone de la función POTENCIA a b, define un procedimiento que la obtenga.

:A para la base
:B para el exponente

```
?PARA LA POTENCIA :A :B
>HAZ "P :A
>REPITE :B - 1 [HAZ "P :P * :A]
>ESCRIBE :P
>FIN
```

Para calcular 6 elevado a 4

```
?POTENCIA 6 4
```

A continuación definimos un procedimiento para dibujar tablas, en el que utilizamos diferentes tipos de expresiones, entre ellas lo que denominamos variables contadoras.

:L para determinar la anchura de la tabla

:D para determinar las divisiones que queremos que tenga la tabla.

```
?PARA TABLA :L :D
>BP
>PONCL 3
>HAZ "A 256 - :L
>HAZ "A :A / 2
>SL
>HAZ "IX - 128 + :A
>PONX :IX
>PONY - 70
>PONRUMBO 0
>BL
>REPITE 2 [AV 140 GD 90 AV :L GD 90]
>SL
>PONX COORX - 1
>PONY COORY - 1
>BL
>REPITE 2 [AV 142 GD 90 AV :L + 2 GD
90]
>SL
>PONX :IX
>PONY 50
>BL PONRUMBO 90
>AV :L
>PONY 51
>RE :L
>SL
>PONY - 70
>HAZ "S :L / :D
>HAZ "C 0
>REPITE :D [PONRUMBO O PONX :IX
+ :C * :S BL AV 120 RE 120
>HAZ "C :C + 1 SL]
>FIN
```

Si la versión de LOGO que utilizas no dispone de una función matemática que necesites, crea un procedimiento para obtenerla.

Por ejemplo, una tabla con 40 de ancho y dos divisiones.

?TABLA 40 2

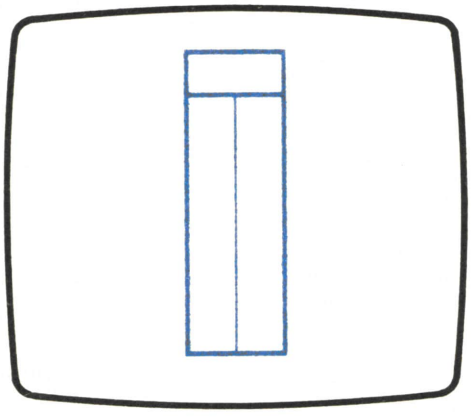


Fig. 1.

también otra con 200 puntos de ancho y cinco franjas.

?TABLA 200 5

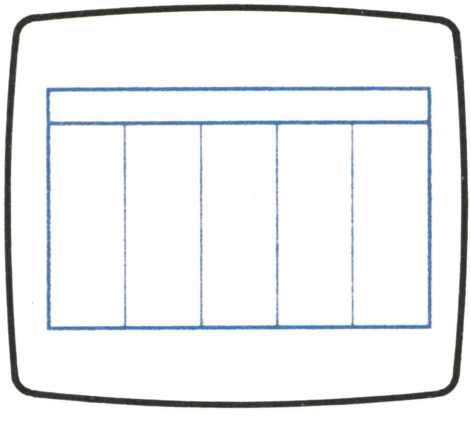


Fig. 2.

Con los siguientes procedimientos conseguimos dibujar maceteros con flores en su interior.

El primer procedimiento que llamamos *macetero,* es el que dibuja el macetero, pudiendo ser éste de la longitud que queramos. Esta longitud la introducimos por medio de la variable :L. En primer lugar, se hace un cálculo para dibujar el macetero centrado en la pantalla. Posicionamos a la Tortuga a una distancia del centro de la pantalla igual a la mitad de la longitud que hemos determinado (:L / 2). Utilizamos la variable :V para guardar este valor. Una vez que la Tortuga está situada en la posición adecuada, comenzamos a dibujar el macetero. Desde este mismo procedimiento se hace una llamada al procedimiento que dibuja las flores.

```
?PARA MACETERO :L :LL :Z
>BP
>SL
>HAZ "V :L / 2
>PONX -:V
>PONRUMBO 90
>BL
>AV :L GI 90
>AV 3 GD 90
>AV 3 GI 90
>AV 10 GD 90
>AV 1 GI 90
>AV 3
>SL
>PONX -:V
>PONY 0
>BL
>AV 3 GI 90
>AV 3 GD 90
>AV 10 GI 90
>AV 1 GD 90
>AV 3 GD 90
>AV :L + 8
>HAZ "W REDONDEA :L / 25
>PONX ( -:W / 2) * 25 + 12
>REPITE :W [FLOR :LL :Z]
>FIN
```

Antes de hacer la llamada al procedimiento Flor, realizamos el cálculo de las flores que caben, que dependerá de la longitud que hayamos dado al macetero. Fijamos en 25 el número de puntos que ocupa cada flor, por lo que, dividiendo la longitud del macetero (:L) por 25, obtenemos las flores que caben. A continuación se hace el cálculo para posicionar a la Tortuga en donde tiene que comenzar a dibujar las flores para que queden éstas centradas. Posicionamos a la Tortuga a una distancia del centro y hacia la izquierda, igual a la mitad del número de flores que caben multiplicado por lo que ocupa cada una de ellas (25) más 12, que corresponde a la mitad de lo que ocupa cada una, aproximadamente.

En la variable :W guardamos el valor correspondiente al número de flores que hay que dibujar, por lo que se realiza una llamada al procedimiento *Flor* tantas veces como indica el valor de :W.

El procedimiento *Flor* utiliza dos variables, una, para determinar la altura de la flor

No olvides dar a una variable un valor inicial.

(:LL), y, otra, para determinar si queremos que la flor se dibuje recta o con una cierta inclinación hacia la izquierda. Para esto utilizamos la variable :Z, que si contiene el valor cero significará que la flor se tiene que dibujar derecha, y si contiene un uno, que se dibuje inclinada. Los valores de estas dos variables son introducidos por el procedimiento macetero.

```
?PARA FLOR :LL :Z
>SL
>PONY 16
>PONRUMBO 0
>BL
>AV :LL / 2
>PONRUMBO -30 * :Z
>AV :LL / 2
>REPITE 8 [HOJA GD 45]
>RE :LL / 2
>GD 45
>HOJA
>PONRUMBO 0
>RE 5
>GI 90
>HOJA
>SL
>PONX COORX + 25
>FIN
```

En este procedimiento, lo primero que hacemos es dibujar el tallo de la flor. Dibujamos la mitad y a continuación ponemos rumbo a la Tortuga hacia -30 * :Z, es decir, que si la variable :Z vale 0, el producto dará 0, por lo que la Tortuga no variará su rumbo y dibujará el tallo derecho. Si, por el contrario, el valor de la variable :Z es igual a 1, el producto dará -30, y la Tortuga variará su rumbo hacia -30 grados. De esta forma el tallo aparecerá inclinado.

En este procedimiento se hace ocho llamadas al procedimiento *Hoja* para dibujar las hojas de la flor que forman el capullo y dos llamadas más para dibujar dos hojas en el tallo.

```
?PARA HOJA
>REPITE 2 [AV 5 GD 45 AV 5 GD 135]
>FIN
```

Si deseamos dibujar un macetero de 25 puntos de longitud y con flores de 30 puntos de largo y derechas, tendremos que ejecutar el procedimiento macetero, introduciendo los siguientes valores a las variables:

?MACETERO 25 30 0

Y obtenemos el siguiente dibujo:

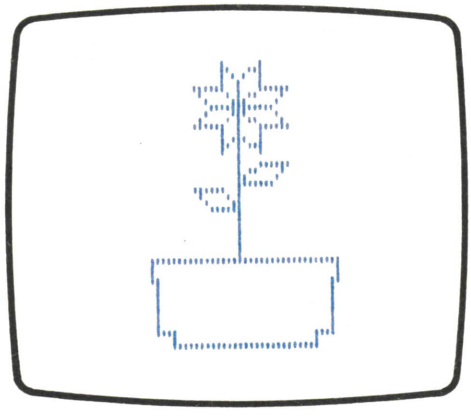


Fig. 3.

Sólo aparece una flor, ya que la longitud del macetero es igual a 25, que es lo que ocupa cada flor.

Si deseamos dibujar un macetero de 100 puntos de longitud, con flores de 50 puntos de altura y derechas, lo ejecutamos de la siguiente forma:

?MACETERO 100 50 0

y en la pantalla obtenemos el siguiente dibujo:

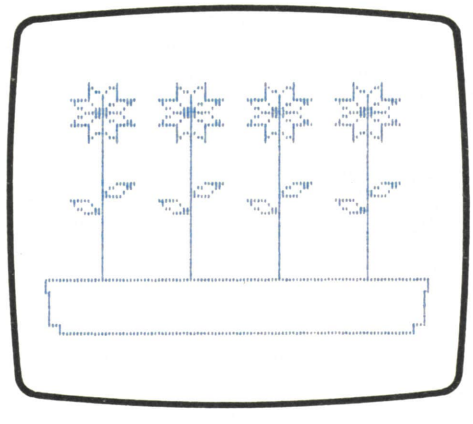


Fig. 4.

Para obtener un macetero de 75 puntos de longitud con flores inclinadas de 40 puntos de altura:

?MACETERO 75 40 1

Utiliza las variables como acumuladores para facilitar tu trabajo.

y obtenemos el siguiente dibujo:

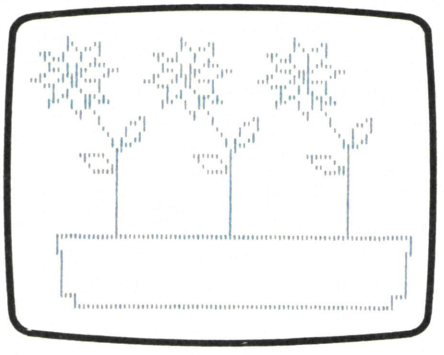


Fig. 5.

Ejercicios

1. *¿Cómo conseguirías obtener los 20 primeros múltiplos de 5? ¿Y los 20 primeros múltiplos de un número dado N?*

2. *Definiendo un procedimiento que dibuje una cara cuadrada y otro para posicionar la primera cara de una tira, intenta realizar este dibujo.*

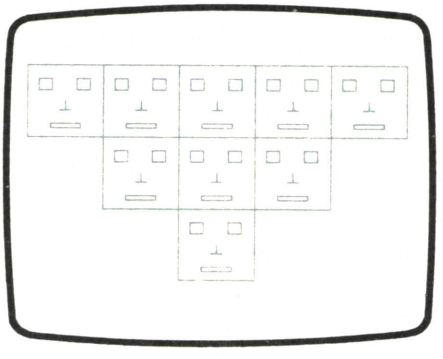


Fig. 6.

3. *Intenta realizar el siguiente dibujo:*

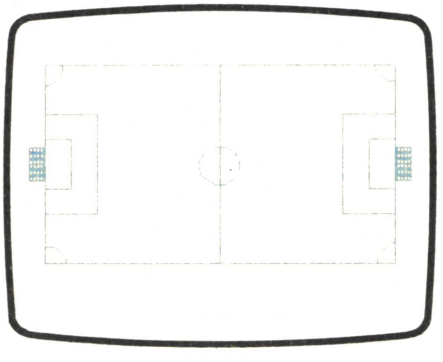


Fig. 7.

4. *Con los procedimientos necesarios intenta dibujar un termómetro con los que se pueda simular la subida del mercurio hasta marcar una temperatura determinada entre los –20 y los +65 grados.*

5. *¿Son correctas las órdenes siguientes?*

```
- ?HAZ "C2
?REPITE :C [HAZ "C :C +1 ESCRIBE :C]

? HAZ "C 0
?HAZ "C (:C + 1) + 1

- ?REPITE 10 [HAZ "C :C + 1 ESCRIBE :C]

- ?PARA A :V
>AV :V
>REPITE :V [AV 10 GD :V]
>B :V + 10
>FIN

- ?PARA A :V
>HAZ "A :W + 1
>FIN
```

Solución a los ejercicios

1: Para obtener los múltiplos de cinco:

```
?PARA MULS
>HAZ "C 0
>REPITE 20 [HAZ "C (:C + 1) HAZ "M (5
* :C) ESCRIBE :M]
>FIN
```

Para obtener los 20 primeros múltiplos de cualquier número N:

```
?PARA MUL :N
>HAZ "C 0
>REPITE 20 [HAZ "C (:C + 1) HAZ "M
(:N * :C) ESCRIBE :M]
>FIN
```

Así, para obtener los múltiplos de 7 ejecutamos el procedimiento MUL, indicando en la llamada el número 7.

```
?MUL 7
```

Los múltiplos de 9.

```
?MUL 9
```

2: Con el procedimiento CARAC dibujamos una cara.

A una variable global le podemos asignar el valor de una local.

```
?PARA CARAC
>HAZ "PX COORX
>HAZ "PY COORY
>BL
>REPITE 4 [AV 40 GD 90]
>SL
>PONX :PX + 6
>PONY :PY + 26
>BL
>REPITE 4 [AV 7 GD 90]
>SL
>PONY COORX + 20
>BL
>REPITE 4 [AV 7 GD 90]
>SL
>PONX :PX + 20
>PONY :PY + 15
>BL
>AV 5 RE 5 GD 90 AV 3 GI 180 AV 6
>PONRUMBO 0
>SL
>PONX :PX + 12
>PONY :PY + 5
>BL
>REPITE 2 [AV 3 GD 90 AV 16 GD 90]
>SL
>PONX :PX
>PONY :PY
>FIN
```

El procedimiento DIBUJO sitúa a la Tortuga en la posición (X Y) y allí empieza a dibujar las N caras.

:X posición en el eje X
:Y posición en el eje Y
:N número de caras seguidas que queremos dibujar

```
?PARA DIBUJO : X :Y :N
>SL
>PONRUMBO 0
>PONX :X
>PONY :Y
>HAZ "C 1
>CARAC
>REPITE (:N - :C) [SL PONX :X + 40 *
:C BL CARAC HAZ "C :C + 1]
>FIN
```

Para conseguir el dibujo que hemos propuesto, tecleamos las órdenes:

```
?BP
?PONCL 2
?DIBUJO -100 40 5
?DIBUJO -60 0 3
?DIBUJO -20 -40 1
```

Ejecutando tres veces DIBUJO conseguimos las tres filas de caras.

3: Para realizar el dibujo de un campo de fútbol, definimos cuatro procedimientos: el primero, y general, el que llamamos CAMPO con el que dibujamos el contorno del campo, y desde el que se realizan las llamadas a los procedimientos que dibujan los puntos de córner, las áreas y las porterías.

```
PARA CAMPO
>BP
>OT
>SL
>PONPOS [-100 -60]
>BL
>REPITE 2 [AV 120 GD 90 AV 200 GD
90]
>PONX 0
>AV 120
>PONX -90
>PONRUMBO 180
>CORNER
>SL
>PONPOS [-100 -50]
>PONRUMBO 90
>CORNER
>SL
>PONPOS [90 - 60]
>PONRUMBO 0
>CORNER
>SL
>PONPOS [100 50]
>PONRUMBO -90
>CORNER
>SL
>PONPOS [-11 0]
>BL
>REPITE 36 [AV 2 GD 10]
>AREAS
>PORTERIAS
>FIN
```

Siempre que necesites hacer varias veces una misma operación con una variable, hazlo una sola vez y guarda el resultado en una variable.

Se hacen cuatro llamadas al procedimiento CORNER para dibujar en cada esquina del campo un pequeño arco que representan los puntos de «saque de esquina»:

```
?PARA CORNER
>BL
>REPITE 9 [AV 2 GD 10]
>FIN
```

A continuación, se hace una llamada al procedimiento AREAS que es el que dibuja las dos áreas que hay en cada lado del campo:

```
?PARA AREAS
>SL
>PONPOS [-100 30]
>PONRUMBO 90
>REPITE 2 [BL AV 30 GD 90 AV 60 GD
  90 AV 30 GD 90 AV 15 GD 90 AV 15
  GI 90 AV 30 GI 90 AV 15 SL PONPOS
  [100 -30]]
>FIN
```

Por último, se realiza una llamada al procedimiento PORTERIAS para dibujar en cada lado del campo una portería:

```
?PARA PORTERIAS
>SL
>PONPOS [-101 10]
>PONRUMBO 180
>BL
>REPITE 2 [REPITE 2 [AV 20 GD 90 AV
  8 GD 90]
  REPITE 3 [GD 90 AV 2 GI 90 AV 20 RE
  20]
  GI 90 RE 2 REPITE 10 [AV 8 RE 8 GD
  90 AV 2 GI 90]
  SL PONPOS [101 -10] PONRUMBO 0
  BL]
>FIN
```

Para ejecutarlo, bastará con introducir el nombre del procedimiento general CAMPO:

```
?CAMPO
```

4: Primero, definimos un procedimiento que dibuja un termómetro. Sobre él dibujamos una serie de rayas que indican la escala de temperaturas. Una que dibujamos más larga, es la que marca los cero grados. En la parte inferior del termómetro dibujamos una zona redonda donde se encuentra el mercurio:

```
?PARA TERMOMETRO
>BP
>SL
>PONPOS [-15 -30]
>BL
>AV 100
>REPITE 9 [AV 2 GD 20]
>AV 102
>GI 65
>REPITE 30 [GD 10 AV 2]
>SL
>PONPOS [-12 - 35]
>PONRUMBO 0
>BL
>AV 100 GD 90 AV 5
>GD 90 AV 100
>GI 65
>REPITE 30 [GD 10 AV 1]
>SL
>PONPOS [-11 -25]
>PONRUMBO 90
>BL
>AV 3
>SL
>GD 135 AV 2
>BL
>RELLENA
>SL
>PONPOS [-4 -20]
>PONRUMBO 90
>REPITE 9 [BL AV 3 RE 3 SL GI 90 AV
10 GD 90]
>SL
>PONPOS [-4 -15]
>REPITE 9 [BL AV 2 RE 2 SL GI 90 AV
10 GD 90]
>SL
>CENTRO
>GD 90
>BL
>AV 3 RE 6
>FIN
```

El procedimiento que llamamos SUBIDA, es el que va a simular el ascenso del mercurio hasta indicar la temperatura deseada. En él utilizamos la variable :T, que es la que va a contener el valor de la temperatura.

Utilizando variables, podemos definir procedimientos para hacer cálculos complicados.

```
?PARA SUBIDA :T
>SL
>PONPOS [-11 - 25]
>PONRUMBO 90
>REPITE :T + 26 [BL AV 3 RE 3 SL GI
90 AV 1 GD 90]
>HAZ "TT :T
>FIN
```

Definimos también el procedimiento BORRAR, con el que podemos hacer que el mercurio descienda hasta su estado inicial, borrando lo que anteriormente se ha dibujado para señalar una temperatura:

```
?PARA BORRAR
>GOMA
>REPITE :TT + 26 [GD 90 AV 1 GI 90
AV 3 RE 3]
>FIN
```

Una vez que tenemos todos los procedimientos necesarios, vamos a ver el modo de ejecutarlos.

Primero, ejecutamos el procedimiento TERMOMETRO:

```
?TERMOMETRO
```

y en la pantalla aparecerá:

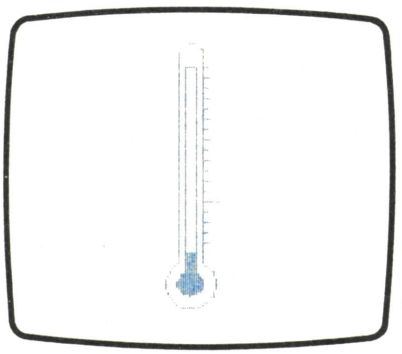


Fig. 8.

Observa que el mercurio está en un estado inicial sin marcar ninguna temperatura. Las rayas que aparecen señalan la temperatura de cinco en cinco grados. Las que son un poco más largas corresponden a los grados desde el -20 hasta 60 señaladas de 10 en 10.

Una vez que tenemos dibujado el termómetro, ejecutamos el procedimiento SUBIDA, donde tenemos que introducir la temperatura que queremos que marque. Si queremos que señale los 0 grados:

```
?SUBIDA 0
```

y en la pantalla aparecerá:

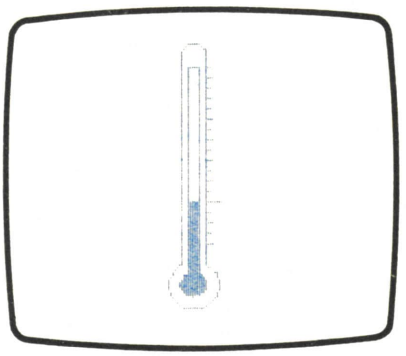


Fig. 9.

Si ahora queremos indicar una temperatura inferior a los 0 grados, tenemos primero que borrar la temperatura anterior. Para estos utilizamos el procedimiento BORRAR. El mercurio bajará hasta su estado inicial:

```
?BORRAR
```

Ahora ya podemos indicar otra temperatura, por ejemplo, -10 grados:

```
?SUBIDA -10
```

El termómetro estará en el siguiente estado:

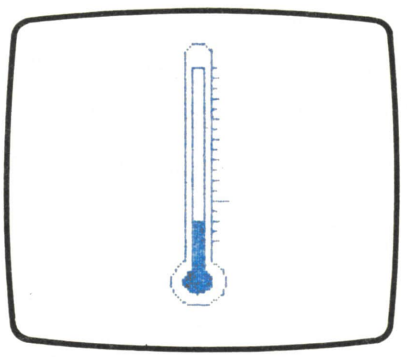


Fig. 10.

Si queremos que señale los 30 grados:

```
?SUBIDA 30
```

Recuerda que el valor asignado a una variable global permanece hasta que no se le haga una nueva asignación.

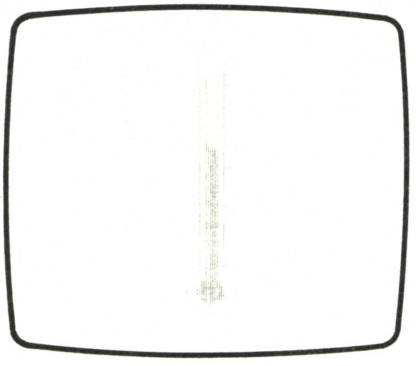


Fig. 11.

5.

- ?HAZ "C 2
?REPITE :C [HAZ "C :C + 1 ESCRIBE :C]
CORRECTO

En el REPITE la variable C vale 2, por tanto, lo que está encerrado entre los corchetes se ejecutará dos veces.

- ?HAZ "C 0
?HAZ "C (:C + 1) + 1
CORRECTO

Sería lo mismo que escribir:

?HAZ "C 0
?HAZ "C :C + 2

- ?REPITE 10 [HAZ "C :C + 1 ESCRIBE :C]
INCORRECTO

No le hemos dado un valor inicial a la variable C. El LOGO nos muestra el mensaje:

C NO TIENE VALOR

- ?PARA A :V
>AV :V
>REPITE :V [AV 10 GD :V]
>B :V + 10
>FIN
CORRECTO

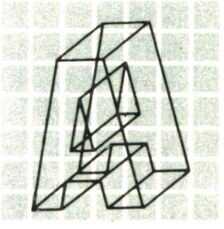
En el procedimiento A se hace una llamada al procedimiento B, que deberá tener una entrada. El valor que pasa a la variable que contenga el procedimiento b, será igual al valor de la variable :v incrementada en 10.

- ?PARA A :V
>HAZ "A :W + 1
>FIN
INCORRECTO

A la variable A se la intenta asignar el valor de la variable :W más 1, pero, al no estar la variable :W inicializada con algún valor, recibimos un mensaje de error.

Podemos asignar a una variable global el resultado de las operaciones de otras variables.

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS



UNQUE hay programas en los que es una sola la figura que se mueve por pantalla, tal es el caso del programa gusano loco, lo normal es que se esté realizando el movimiento de dos o más figuras simultáneamente. La verdad es que dicho movimiento no es nunca simultáneo, pero la rapidez con que trabaja nuestro ordenador nos hace verlos como movimientos simultáneos.

En los programas comerciales, escritos casi todos en código máquina, se suelen utilizar las interrupciones del microprocesador para realizar el movimiento de algunas de las figuras, para tocar música. Algunos ordenadores, de los que aquí estamos viendo, poseen la capacidad de trabajar con interrupciones desde el BASIC. Estos ordenadores son el IBM, el AMSTRAD y el MSX. El SPECTRUM y el COMMODORE no las tienen incorporadas. Es por ello que realizaremos el estudio del movimiento de varias figuras por pantalla sin tener en cuenta las interrupciones. Más adelante, en sucesivos tomos, hablaremos más sobre ellas.

Movimiento de varias figuras animadas

Cualquiera de los ya famosos programas de ordenador tiene que utilizar rutinas de movimiento que controlen más de una figura a la vez por la pantalla. El PAC-MAN, las moscas, los marcianitos, todos ellos tienen en común que muchas figuras se están moviendo a la vez por la pantalla.

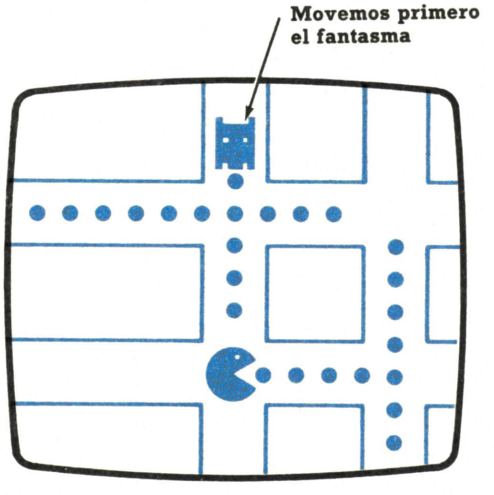


Fig. 1. Hay que mover una figura detrás de otra. Si lo hacemos lo suficientemente buen se

Realizar dichas rutinas de movimiento son muy sencillas, su forma es igual que la que hemos visto hasta ahora. La única diferencia importante se encuentra cuando tenemos que saber a qué figura le toca moverse. Como regla, casi general, podemos decir que el movimiento de las figuras va a ser secuencial. Esto significa que, tras la figura número uno, siempre se moverá la número dos y tras ésta la número tres. Si el ordenador es lo suficientemente rápido, no se notará ninguna pausa en el movimiento de ninguna de las figuras.

El primer programa que vamos a ver nos ilustra un poco más claramente sobre el tema.

```
100 REM ***********************************
110 REM * MOVIMIENTO SIMULTANEO DE VARIAS *
120 REM * FIGURAS POR LA PANTALLA         *
130 REM ***********************************
140 REM
150 CLS
160 REM
170 REM *** POSICION INICIAL DE LA 'O' ***
180 REM
190 LET XO=1
200 LET YO=1
210 LET X1=1
220 LET Y1=1
230 LET S1=1
240 REM
250 REM *** POSICION INICIAL DE LA 'X' ***
260 REM
270 LET XX=15
280 LET YX=20
290 LET X2=15
300 LET Y2=20
310 LET S2=1
320 REM
330 REM *** POSICION INICIAL DEL '*' ***
340 REM
350 LET XA=20
360 LET YA=20
370 LET X3=20
380 LET Y3=20
390 LET S3=1
400 REM
410 REM *** PROGRAMA PRINCIPAL
420 REM
430 GOSUB 1000 : REM MOVIMIENTO DE LA 'O'
440 GOSUB 2000 : REM MOVIMIENTO DE LA 'X'
450 GOSUB 3000 : REM MOVIMIENTO DEL '*'
460 LET A$=INKEY$
470 IF A$="P" THEN GOTO 490
480 GOTO 430
490 REM
500 REM *** DESPEDIDA ***
510 REM
520 CLS
530 PRINT "ADIOS ..."
540 FOR I=1 TO 10
550    PRINT
560 NEXT I
570 END
1000 REM
1010 REM *** MOVIMIENTO DE LA 'O' ***
1020 REM
1030 LOCATE Y1,X1
1040 PRINT " "
1050 LET XO=XO-(S1=1)+(S1=3)
1060 LET YO=YO-(S1=2)+(S1=4)
1070 ON S1 GOSUB 1130,1150,1170,1190
1080 LOCATE YO,XO
1090 PRINT "O"
1100 LET Y1=YO
1110 LET X1=XO
1120 RETURN
1130 IF XO=15 THEN LET S1=2
1140 RETURN
1150 IF YO=10 THEN LET S1=3
1160 RETURN
```

```
1170 IF XO=1 THEN LET S1=4
1180 RETURN
1190 IF YO=1 THEN LET S1=1
1200 RETURN
2000 REM
2010 REM *** MOVIMIENTO DE LA 'X' ***
2020 REM
2030 LOCATE Y2,X2
2040 PRINT " "
2050 LET YX=YX-(S2=2)+(S2=1)
2060 LET S2=-(S2=2 AND YX=20)-2*(S2=1 AND YX=1)-S2*(YX<>1 AND YX<>20)
2070 LOCATE YX,XX
2080 PRINT "X"
2090 LET X2=XX
2100 LET Y2=YX
2110 RETURN
3000 REM
3010 REM *** MOVIMIENTO DEL '*' ***
3020 REM
3030 LOCATE Y3,X3
3040 PRINT " "
3050 LET XA=XA-(S3=2)+(S3=1)
3060 LET S3=-(S3=2 AND XA=30)-2*(S3=1 AND XA=1)-S3*(XA<>1 AND XA<>30)
3070 LOCATE YA,XA
3080 PRINT "*"
3090 LET X3=XA
3100 LET Y3=YA
3110 RETURN
```

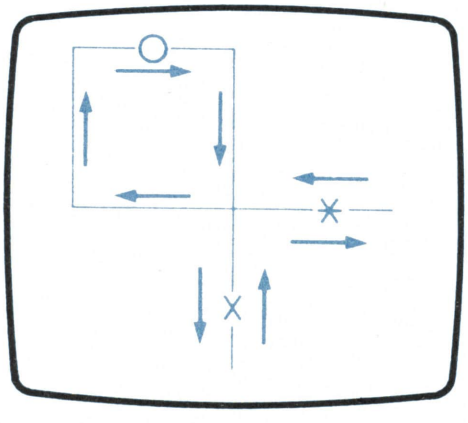


*Fig. 2. Con el programa 0521 podremos ver a una O mayúscula moverse siguiéndo un cuadro, a una X moviendose de arriba a abajo y a un * moviéndose de derecha a izquierda.*

El programa nos muestra por pantalla a una 'O' que se mueve sobre un cuadrado, una 'X' moviéndose de arriba a abajo (y de abajo a arriba) y un asterisco (*) moviéndose de izquierda a derecha (y de derecha a izquierda), simultáneamente.

Se puede apreciar que hay momentos en que los distintos caracteres, al moverse por la pantalla, chocan entre sí, aunque sin ocasionar ninguna interferencia en su movimiento.

Las variaciones que son necesarias hacer para que el programa funcione en ordenadores que no sean IBM o compatibles son las siguientes:

COMODORE

```
 150 PRINT CHR$ (147)
 460 GET A$
 520 PRINT CHR$ (147)
1030 POKE 214, Y1: POKE 211, X1
1060 POKE 214, YO: POKE 211, XO
2030 POKE 214, Y2: POKE 211, X2
2060 POKE 214, YX: POKE 211, XX
3030 POKE 214, Y3: POKE 211, X3
3060 POKE 214, YA: POKE 211, XA
```

SPECTRUM

```
 570 STOP
1030 PRINT AT Y1, X1;
1050 GOSUB 1110* (S1=1)+1130*(S1=2)
+1150*(S1=3)+1170*(S1=4)
1060 PRINT AT YO, XO;
2030 PRINT AT Y2, X2;
2050 GOSUB 2110*(S2=1)+2130*(S2=2)
2060 PRINT AT YX, XX;
3030 PRINT AT Y3, X3;
3050 GOSUB 3110*(S3=1)+3130*(S3=2)
```

AMSTRAD Y MSX

```
1030 LOCATE X1, Y1
1060 LOCATE XO, YO
2030 LOCATE X2, Y2
2060 LOCATE XX, YX
3030 LOCATE X3, Y3
3060 LOCATE XA, YA
```

Una vez que el usuario se canse de ver cómo se mueven las figuras por la pantalla, sólo necesita pulsar la 'P' (mayúscula) para que la demostración termine.

El funcionamiento del programa, línea a línea, es el siguiente:

Línea 150. Borramos la pantalla.

Línea 190. Asignamos a la variable XO el valor uno (1). Esta variable será la encargada de ir almacenando la posición en X de la 'O' al moverse por la pantalla.

Línea 200. Asignamos a la variable YO el valor uno (1). Esta se encargará de almacenar la posición en Y del movimiento de la 'O' por la pantalla.

Línea 210. Asignamos a la variable X1 el valor uno (1). Esta variable se encargará de almacenar la antigua posición en X del movimiento de la 'O'.

Línea 220. Asignamos a la variable Y1 el valor uno (1). Esta variable se encargará de almacenar la antigua posición en Y del movimiento de la 'O'.

Línea 230. Asignamos a la variable S1 el valor uno (1). Esta variable será la que nos indique en qué dirección se ha de mover la 'O'. Según valga 1, 2, 3 o 4, la 'O' se moverá en una dirección diferente, según la siguiente tabla:

Valor 1: Movimiento hacia la derecha
Valor 2: Movimiento hacia abajo.
Valor 3: Movimiento hacia la izquierda.
Valor 4: Movimiento hacia arriba.

Línea 270. Almacenamos en la variable XX el valor 15. Esta variable nos dirá la posición en X durante el movimiento de la 'X' por la pantalla.

Línea 280. Asignamos a la variable YX el valor 20. Esta variable nos dirá la posición en Y del movimiento de la 'X' por la pantalla.

Línea 290. Asignamos a la variable X2 el valor 15. Esta variable será la encargada de almacenar la última posición en X de la 'X' en la pantalla.

Línea 300. Almacenamos en la variable Y2 el valor 20. Esta variable almacenará la última posición en Y de la 'X' en la pantalla.

Línea 310. Almacenamos en la variable S2 el valor uno (1). Esta variable nos dirá la dirección del movimiento de la 'X' por la pantalla según la siguiente tabla:

Valor 1: Movimiento hacia arriba.
Valor 2: Movimiento hacia abajo.

Línea 350. Almacenamos en la variable YA el valor 20. Esta variable nos informará de la posición en X del asterisco (*) en la pantalla.

Línea 360. Asignamos a la variable YA el valor 20. Esta nos indicará la posición en Y del asterisco (*).

Línea 370. Asignamos a la variable X3 el valor 20. Esta variable almacenará la última posición en la pantalla del asterisco.

Línea 380. Almacenamos en la variable Y3 el valor 20. Esta variable será la encargada de almacenar la última posición en Y del asterisco en la pantalla.

Línea 390. Asignamos a la variable S3 el valor uno (1). Esta variable nos indicará el sentido del movimiento del asterisco por la pantalla, según la siguiente tabla:

Valor 1: Hacia la izquierda.
Valor 2: Hacia la derecha.

La definición de todas estas variables son necesarias para el buen funcionamiento del programa. Se ha hecho hincapié en ellas para que se pueda entender mejor cómo funciona el programa. Como se puede apreciar, los nombres de las variables no son arbitrarios. Se han puesto los nombres de forma que la propia variable nos diga qué carácter controla. Así, las variables XO y YO controlan el movimiento en X y en Y de la 'O'. Las variables XX y YX el movimiento de la 'X'. Y las variables XA y YA el movimiento del Asterisco (*). Es una buena costumbre hacer que el nombre de las variables nos recuerden la función para la que sirven.

Por otro lado, las variables X1, Y1, X2, Y2, X3 y Y3 son variables auxiliares que almacenarán la antigua posición de los caracteres a mover. En este caso, el nombre de las variables no tienen nada que ver con la función que realizan. Esto es porque a la hora de realizar el programa, como se intenta que sea lo más estándar posible para que sirva para todos los ordenadores, se ha tenido que restringir el nombre de las variables a dos caracteres.

Por fin, y ya para terminar con la definición de las variables, se han definido una serie de SWICH (interruptores) que nos indicarán la dirección del movimiento de cada uno de los caracteres, según su valor. El nombre de estas variables tampoco coincide plenamente con la función que realizan por la razón explicada más arriba. Aun así, se ha intentado que haya una relación entre éstas y las variables auxiliares.

switch = interruptor

Este tipo de variable se les llama Swich porque solo pueden tener unos ciertos valores prefijados que se comportan como interruptores de dos o más posiciones. Los valores más normales que suelen tener son:

Ø—apagado
1—encendido

A continuación pasamos a explicar el funcionamiento del corazón del programa.

Línea 430. Llamamos a la rutina que empieza en la línea 1.000. Esta rutina se dedica a gestionar el movimiento de la 'O' por la pantalla.

Línea 440. Llamamos a la rutina que empieza en la línea 2.000 y que se encarga de mover la 'X' por la pantalla.

Línea 450. Llamamos a la rutina que está localizada en la línea 3.000 y siguientes y que mueve el asterisco por la pantalla.

Línea 460. Miramos si el usuario ha pulsado alguna tecla.

Línea 470. Si la tecla pulsada es una 'P' mayúscula entonces nos vamos a la línea **490** para terminar el programa.

Línea 480. Volvemos a la línea **430** mediante un GOTO.

Esta es la forma general que ha de tener un programa que se encarga de gestionar el movimiento de varias figuras por la pantalla. Hay otras maneras de hacerlo que nos permiten ahorrar memoria y ganar velocidad, pero primero es necesario entender el funcionamiento de este tipo de estructura. Más adelante veremos otras formas más avanzadas de control de movimiento, creando una sola rutina para gestionar todos los movimientos en la pantalla.

Las líneas que se encuentran enre la **490** y la **570** son, la despedida del programa. Debido a su sencillez, la explicación no se hace necesaria.

La parte del programa que nos queda por estudiar podemos dividirlas en tres trozos distintos, atendiendo al carácter que gestiona cada parte del programa. Es por ello que cada rutina empieza en un número de línea que se distingue claramente de los demás. Veamos una a una dichas rutinas, empezando por la que controla el movimiento de la 'O'.

Línea 1030. Colocamos el cursor en la posición que nos indican las variables X1 e Y1 para borrar la 'O' de su posición actual. La primera vez que se ejecute la rutina no habrá nada que borrar porque no hay nada impreso en la pantalla, pero a partir de la segunda, borraremos la 'O' de la pantalla momentos antes de imprimirla en la siguiente posición que le corresponda.

Línea 1040. Imprimimos un carácter en blanco, o lo que es igual, borramos la 'O'.

Línea 1050. Según el valor de la variable S1 nos vamos a un número de línea determinado. Ya veremos qué se hace en dichos números de líneas. Lo único que tenemos que saber, por el momento, es que, cuando la rutina seleccionada devuelva el control a esta parte del programa, las variables YO y XO tendrán la nueva posición en la pantalla de la 'O'.

Línea 1060. Colocamos el cursor en la posición que acabamos de hallar.

Línea 1070. E imprimimos la famosa 'O'.

Líneas 1080 y **1090.** Almacenamos en las variables X1 e Y1 el valor de XO e YO, respectivamente. Almacenamos lo que dentro de unos momentos va a ser la antigua posición de la 'O' en pantalla. Gracias a estas asignaciones podremos saber dónde está la 'O' para poder borrarla.

Línea 1100. Devolvemos el control al programa principal.

Línea 1110. En el caso de que S1 sea igual a uno (1), la línea **1050,** transferirá el control del programa a esta línea. En ella sumamos uno a la posición en X de la 'O' para moverla una posición a la derecha. Si hemos llegado al límite permitido (si XO es igual a 15) entonces hacemos que S1 sea igual a dos (2) para que la próxima vez que se tenga que mover la 'O' lo haga hacia abajo y no hacia la derecha.

Línea 1120. Devolvemos el control del programa a la línea **1060** para dibujar la 'O' en su nueva posición.

Línea 1130. Realizamos casi lo mismo que se hizo en la línea **1110,** cuando S1 era igual a uno (1), pero ahora, como el movimiento es hacia abajo, tenemos que sumarle uno a la posición en Y de la 'O'. Si ésta ha llegado al límite permitido, significa que es hora de hacer que el movimiento sea hacia la izquierda, por lo que el valor de S1 se pone a tres (3).

Línea 1140. Nos volvemos a la línea **1060.**

Línea 1150. Lo mismo que hemos visto anteriormente. Hacemos que varíe la componente en X de la 'O' y cuando ha llegado al tope cambiamos el valor de S1 para que tenga valor cuatro (4), y movernos hacia arriba.

Línea 1160. Nos volvemos a la línea **1060.**

Línea 1170. También esta línea es muy semejante a las anteriores, pero con una pequeña e importante diferencia. Cuando el valor de TO es igual a uno (1) significa que la 'O' ya ha completado la trayectoria de un cuadrado. En ese momento el movimiento ha de ser de nuevo hacia la derecha, por eso hacemos que S1 vuelva a tener valor uno (1). Gracias a esto el movimiento es infinito hasta que se pulse la 'P'.

Línea 1180. Nos volvemos a la línea **1060.**

Se puede apreciar que la estructura de esta rutina es muy repetitiva. Se podía haber hecho mucho más corta, pero mucho más difícil de entender. Se propone al lector que intente variarla para que gane en velocidad y eficiencia.

La siguiente rutina, muy similar a la que acabamos de ver, se encarga de mover una 'X' de arriba a abajo y de abajo a arriba por la pantalla. Dicha rutina es la que empieza en la línea **2000.** No vamos a explicar su funcionamiento línea a línea, pues el lector está perfectamente capacitado para poder entenderlo. Tan sólo hay que decir que, como el movimiento sólo puede tener dos sentidos (arriba y abajo), el valor de S2 sólo puede tomar valor uno y dos. Por lo demás, la filosofía del funcionamiento es la misma.

Por fin, la tercera y última rutina del programa (la que empieza en la línea **3000**) funciona igual que la anterior, pero moviendo un asterisco de izquierda a derecha y de derecha a izquierda. En este caso la variable S3 sólo puede tener dos valores, uno y dos.

Un poco más arriba, en el texto, se propuso al lector intentar variar las rutinas vistas para que sean más veloces y nos permitan ahorrar más memoria y hagan de nuestro programa algo bonito y reflejo de una idea más feliz. La solución que aparece a continuación es una de las muchas que hay, y, por supuesto, no es la mejor. Seguro que algún usuario ha encontrado otra más bonita.

```
100 REM ************************************
110 REM * MOVIMIENTO SIMULTANEO DE VARIAS *
120 REM * FIGURAS POR LA PANTALLA         *
130 REM ************************************
140 REM
150 CLS
160 REM
170 REM *** POSICION INICIAL DE LA 'O' ***
180 REM
190 LET XO=1
200 LET YO=1
210 LET X1=1
220 LET Y1=1
230 LET S1=1
240 REM
250 REM *** POSICION INICIAL DE LA 'X' ***
260 REM
270 LET XX=15
280 LET YX=20
290 LET X2=15
300 LET Y2=20
310 LET S2=1
320 REM
330 REM *** POSICION INICIAL DEL '*' ***
```

```
 340 REM
 350 LET XA=20
 360 LET YA=20
 370 LET X3=20
 380 LET Y3=20
 390 LET S3=1
 400 REM
 410 REM *** PROGRAMA PRINCIPAL
 420 REM
 430 GOSUB 1000 : REM MOVIMIENTO DE LA 'O'
 440 GOSUB 2000 : REM MOVIMIENTO DE LA 'X'
 450 GOSUB 3000 : REM MOVIMIENTO DEL '*'
 460 LET A$=INKEY$
 470 IF A$="P" THEN GOTO 490
 480 GOTO 430
 490 REM
 500 REM *** DESPEDIDA ***
 510 REM
 520 CLS
 530 PRINT "ADIOS ..."
 540 FOR I=1 TO 10
 550    PRINT
 560 NEXT I
 570 END
1000 REM
1010 REM *** MOVIMIENTO DE LA 'O' ***
1020 REM
1030 LOCATE Y1,X1
1040 PRINT " "
1050 ON S1 GOSUB 1110,1130,1150,1170
1060 LOCATE YO,XO
1070 PRINT "O"
1080 LET Y1=YO
1090 LET X1=XO
1100 RETURN
1110 LET XO=XO+1:IF XO=15 THEN LET S1=2
1120 RETURN
1130 LET YO=YO+1:IF YO=10 THEN LET S1=3
1140 RETURN
1150 LET XO=XO-1:IF XO=1 THEN LET S1=4
1160 RETURN
1170 LET YO=YO-1:IF YO=1 THEN LET S1=1
1180 RETURN
2000 REM
2010 REM *** MOVIMIENTO DE LA 'X' ***
2020 REM
2030 LOCATE Y2,X2
2040 PRINT " "
2050 ON S2 GOSUB 2110,2130
2060 LOCATE YX,XX
2070 PRINT "X"
2080 LET X2=XX
2090 LET Y2=YX
2100 RETURN
2110 LET YX=YX-1:IF YX=1 THEN LET S2=2
2120 RETURN
2130 LET YX=YX+1:IF YX=20 THEN LET S2=1
2140 RETURN
3000 REM
3010 REM *** MOVIMIENTO DEL '*' ***
3020 REM
3030 LOCATE Y3,X3
3040 PRINT " "
3050 ON S3 GOSUB 3110,3130
3060 LOCATE YA,XA
```

```
3070 PRINT "*"
3080 LET X3=XA
3090 LET Y3=YA
3100 RETURN
3110 LET XA=XA-1:IF XA=1 THEN LET S3=2
3120 RETURN
3130 LET XA=XA+1:IF XA=30 THEN LET S3=1
3140 RETURN
```

Las modificaciones que hay que realizar para que el programa funcione en ordenadores como el COMMODORE, AMSTRAD, SPECTRUM y MSX son las siguientes:

COMMODORE

```
 150 PRINT CHR$(147)
 460 GET A$
 520 PRINT CHR$(147)
1030 POKE 214,Y1:POKE 211,X1
1080 POKE 214,YO:POKE 211,XO
2030 POKE 214,Y2:POKE 211,X2
2070 POKE 214,YX:POKE 211,XX
3030 POKE 214,Y3:POKE 211,X3
3070 POKE 214,YA:POKE 211,XA
```

SPECTRUM

```
 570 STOP
1030 PRINT AT Y1,X1;
1050 LET XO=XO+(S1=1)-(S1=3)
1060 LET YO=YO+(S1=2)-(S1=4)
1080 PRINT AT YO,XO;
2030 PRINT AT Y2,X2;
2050 LET YX=YX+(S2=2)-(S2=1)
2060 LET S2=(S2=2
     AND YX=20)+2*(S2=1
     AND YX=1)+S2*(YX<>1
     AND YX<>20)
2070 PRINT AT YX,XX;
3030 PRINT AT Y3,X3;
3050 LET XA=XA+(S3=2)-(S3=1)
3060 LET S3=(S3=2
     AND XA=30)+2*(S3=1
     AND XA=1)+S3*(XA<>1
     AND XA<>30)
3070 PRINT AT YA, XA;
```

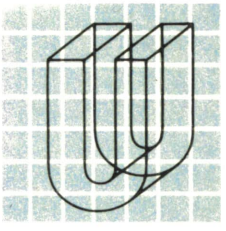
AMSTRAD Y MSX

```
1030 LOCATE X1,Y1
1080 LOCATE XO,YO
2030 LOCATE X2,Y2
2070 LOCATE XX,YX
3030 LOCATE X3,Y3
3070 LOCATE XA,YA
```

Lo que hemos visto en este tomo nos puede dar una idea de cómo gestionar el movimiento de más de una figura a la vez por la pantalla. De todas maneras, hay que decir que cada programa necesita su propia solución específica. Así, un programa de marcianitos necesitaría tres rutinas de movimiento: una, para las naves enemigas; otra, para la nave nodriza, y, otra, para nuestra nave, pero un programa estilo PACMAN se podría realizar con una o dos rutinas distintas de movimiento, dependiendo de la velocidad de nuestro ordenador.

En el siguiente tomo veremos un programa en el cual conjugaremos el control del movimiento de nuestras figuras con el movimiento de otras figuras distintas. En él veremos una solución a un problema específico.

Volcados de memoria



UNA de las cosas más interesantes que podemos hacer con nuestro ordenador es introducirnos en su memoria, introducirnos en el sistema e intentar comprender y aprender su funcionamiento. Aprender código máquina (ensamblador) es una de las labores más difíciles, pero más bonitas y atractivas que un futuro programador ha de llevar a cabo.

Aprendiendo código máquina podremos hacer que nuestro ordenador haga cualquier cosa. No hay límite para las cosas que se pueden hacer. Es verdad que todo lo que se puede hacer en código máquina también se puede realizar en BASIC o en PASCAL o en cualquier otro lenguaje, pero, en la mayoría de las veces, el programa resultante puede ser tan largo y lento que no merezca la pena su realización.

Pero trabajar en código máquina es difícil y lleva muchas horas de estudio y entrenamiento (aunque los resultados pueden ser tan espectaculares que merece la pena). Por ello, todos los programadores que trabajan en código máquina, se ayudan de una serie de programas para hacer más fácil la programación. Entre estos programas se encuentran los DUMPS de memoria.

Un DUMP de memoria no es más que un volcado de memoria, pero, ¿qué es un volcado de memoria? Volcar la memoria significa leer los números que se encuentran almacenados en una serie de posiciones de memoria e imprimirlos en pantalla o en impresora. A primera vista parece que esto no sirve para nada, pero no es cierto. Dichos DUMPS de memoria suelen ser en hexadecimal, esto es, los números que vamos imprimiendo en pantalla, según vamos recorriendo la memoria, se imprimen en hexadecimal. Para un buen programador poder entender lo que significan estos números en hexadecimal es de vital importancia.

El primer programa que vamos a ver a continuación nos permitirá realizar DUMPS (volcados) de memoria en la pantalla o en la impresora. El programa nos imprimirá la dirección de memoria que estamos mirando, el número (en hexadecimal) que hay dentro de dicha posición de memoria, el mismo número en decimal y el carácter que tiene como código ASCII dicho número. Veamos separadamente la función e importancia de cada uno de estos datos.

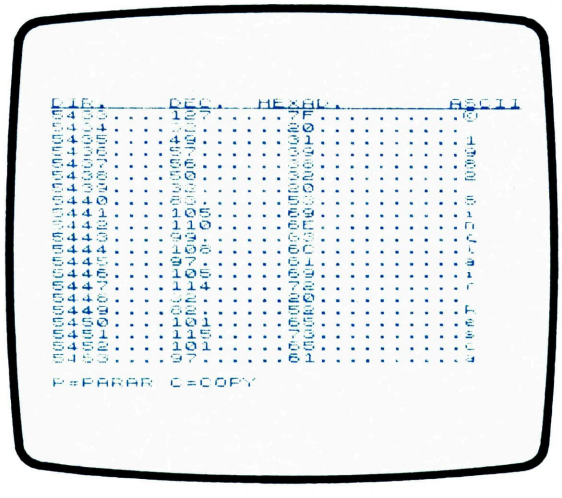


Fig. 1. Ejemplo de DUMP de memoria en el SPECTRUM.

Dirección de memoria. Este número nos va a decir de qué dirección de memoria estamos hablando. Aunque parezca arbitraria la explicación, no lo es. Si nosotros estamos testeando parte del sistema de la máquina o bien un programa o unos datos que nosotros hemos introducido, es muy importante saber

en qué dirección de memoria nos encontramos. Saber qué casilla de memoria es la que estamos mirando. El programa nos da esta dirección en decimal, aunque a veces nos sería más útil poder verlo en hexadecimal. Se puede variar el programa con mucha facilidad para que lo haga.

Número almacenado en hexadecimal. El programa también nos imprime el valor (número) que se encuentra almacenado en dicha posición de memoria. El número aparece en hexadecimal porque muchas veces, para un programador experimentado, un número hexadecimal dice muchas más cosas sobre sí mismo que si estuviese en decimal. Por ejemplo, el número 255, que nos parece un número muy normal, en hexadecimal se escribe FF. Esto significa que la posición de memoria que contiene este número tiene todos los BITs puestos a 1. Esto puede parecer normal, pero el número 15, que en hexadecimal es OF, nos dice que los cuatro BITs de la derecha de dicho valor están puestos a 1 el resto a 0. Por eso se utiliza muchas veces el sistema hexadecimal de numeración, porque con una sola mirada al número nos dice muchas cosas sobre él, sobre su estructura y representación en la memoria.

Número almacenado en decimal. Como no siempre es mejor ver un número en hexadecimal, el programa también nos imprime el contenido de todas las posiciones de memoria en decimal. Si estamos testeando una parte de la memoria y nos encontramos que en cierta posición hay un 3A (en hexadecimal), si imprimimos este número en decimal (es igual a 58), entonces podemos comprender mejor el significado de este número. Para ello, tiene que haber una razón. Por ejemplo, dicho número podría ser la edad de una persona que tenemos almacenada en memoria. En este caso es estúpido y más difícil intentar entender este número en hexadecimal, ya que es un dato y no nos interesa para nada la estructura binaria que pueda tener.

Carácter con dicho código ASCII. Al igual que acabamos de ver al explicar el porqué representamos los números en decimal, hay veces en que un número no nos dice absolutamente nada. Este caso se puede dar si estamos mirando una zona de memoria donde están almacenadas una serie de palabras. En ese caso nos interesa saber el mensaje y nada más. Si nos encontrásemos con una secuencia de números como la siguiente:

72 111 108 97

ésta no nos dice absolutamente nada, pero si imprimimos los caracteres que tienen estos códigos ASCII nos aparecerá por pantalla:

H o l a

mensaje que entendemos a la perfección. Cuando el código ASCII es menor de 32 no se imprime. Esto es debido a que los 31 primeros caracteres ASCII son caracteres de control que se suelen encargar de borrar la pantalla, mover el cursor, borrar caracteres o líneas, etcétera. En este caso no se imprimen porque podrían destrozar el formato del DUMP en la pantalla. En su caso se imprimirá siempre un asterisco.

```
100 REM ******************************
110 REM *      DUMP DE MEMORIA       *
120 REM * PARA TODOS LOS ORDENADORES *
130 REM *    EXCEPTO EL SPECTRUM     *
140 REM ******************************
150 REM
160 REM ******************************
170 REM * POR Fco. Morales Guerrero *
180 REM ******************************
190 REM
200 REM ******************************
210 REM * (c) Ed. Siglo Cultural    *
220 REM * (c) 1987                 *
230 REM ******************************
240 REM
250 DEF FNP$=MID$(A$,1+INT(PP/16),1)+MID$(A$,1+PP-INT(PP/16)*16,1)
260 LET A$="0123456789ABCDEF"
270 CLS
280 LOCATE 10,1
```

```
290 INPUT "Desde la direccion ... ";D1
300 LOCATE 12,1
310 INPUT "Hasta ... ";D2
320 IF D2<D1 OR D1<1 OR D2>65535! THEN GOTO 270
330 FOR I=D1 TO D2 STEP 20
340     CLS
350     PRINT " DIR.   DECIMAL    HEXADECIMAL    ASCII"
360     PRINT "-----------------------------------------"
370     FOR J=I TO I+19
380         LET PP=PEEK(J)
390         LET P$=FNP$
400         IF PP<32 THEN LET C$="*":GOTO 420
410         LET C$=CHR$(PP)
420         PRINT USING "##### ... ###";J;PP;:PRINT " ....... ";P$;" .......... ";
C$
430     NEXT  J
440 PRINT
450 PRINT "(C) COPIAR                    (P) PARAR";
460 LET B$=INKEY$
470 IF B$="" THEN GOTO 460
480 IF B$="C" OR B$="c" THEN GOSUB 530
490 IF B$="P" OR B$="p" THEN GOTO 510
500 NEXT I
510 CLS
520 END
530 REM
540 REM *** SALIDA POR IMPRESORA ***
550 REM
560 LPRINT " DIR.   DECIMAL    HEXADECIMAL    ASCII"
570 LPRINT "-----------------------------------------"
580 FOR J=I TO I+19
590     LET PP=PEEK(J)
600     LET P$=FNP$
610     IF PP<32 THEN LET C$="*":GOTO 630
620     LET C$=CHR$(PP)
630     LPRINT USING "##### ... ###";J;PP;:LPRINT " ....... ";P$;" .......... ";C
$
640 NEXT  J
650 RETURN
```

El programa número uno está preparado para que funcione en todos los ordenadores menos en el SPECTRUM. Para el SPECTRUM hemos preparado el programa número 2.

```
 10 POKE 23658,8
 20 INPUT "DIRECCION INICIO ?";DI
 30 FOR A=DI TO 65535 STEP 21
 40 CLS
 50 PRINT "DIR.    DEC.  HEXAD.       ASCII";AT 0,0; OVER 1;"_________________
_____________"; OVER 0
 60 FOR B=0 TO 20
 70 LET PD=PEEK (A+B)
 80 LET P1=PD/16
 90 LET P=INT P1
100 LET LO=LEN STR$ (A+B)
110 PRINT STR$ (A+B)+("....." AND LO=1)+("...." AND LO=2)+("..." AND LO=3
)+("..." AND LO=4)+("..." AND LO=5);
115 PRINT STR$ (PD)+"......."( TO 7-(PD>9)-(PD>99));
120 PRINT (STR$ P AND P<10)+("A" AND P=10)+("B" AND P=11)+("C" AND P=12)+("D" A
ND P=13)+("E" AND P=14)+("F" AND P=15);
130 LET P2=PD-P*16
140 LET P=P2
150 PRINT (STR$ P AND P<10)+("A" AND P=10)+("B" AND P=11)+("C" AND P=12)+("D" A
ND P=13)+("E" AND P=14)+("F" AND P=15);
```

```
160 PRINT ".........";(CHR$ PD AND PD>31 AND PD<165)+("**" AND (PD<32 OR PD>14
4))
170 NEXT B
180 PRINT #1; FLASH 1;"P=PARAR C=COPY"
190 PAUSE 0
200 LET K$=INKEY$
210 IF K$="P" THEN  CLEAR : GO TO 20
220 IF K$="C" THEN  COPY
230 NEXT A
```

El programa número uno se realizó en un IBM, para utilizarlo en otros ordenadores hay que realizar las siguientes modificaciones:

COMMODORE

```
270 PRINT CHR$(147)
280 POKE 214,9:POKE 211,0
300 POKE 214,11:POKE 211,0
340 PRINT CHR$(147)
420 PRINT J; TAB(10); PP; TAB(22); P$; TAB(35); C$
460 GET B$
510 PRINT CHR$(147)
555 OPEN 1,4
557 CMD 1
560 PRINT "DIR. DECIMAL HEXADECIMAL ASCII"
570 PRINT "----------------------------------------"
630 PRINT J; TAB(10); PP; TAB(22); P$; TAB(35); C$
645 CLOSE 1
```

AMSTRAD Y MSX

```
280 LOCATE 1,10
300 LOCATE 1,12
```

El funcionamiento del programa número uno (todas las versiones, excepto SPECTRUM), línea a línea, es el siguiente:

Línea 250. Definimos una función que llamaremos P$ y que se encargará de pasar un número en decimal a hexadecimal. Lo único que hace esta función es dividir el número entre 16 y quedarse con la parte entera. Una vez que tenemos este primer valor, nos vamos al string A$ (definido en la línea siguiente) y localizamos el carácter que está colocado en la posición de dicho número. Por ejemplo, si el número a pasar era el 200, al coger la parte entera de 200, dividido entre 16, nos da, 12. Si nos vamos al carácter número doce, almacenado en A$, éste resulta ser una C. Esta C (12) es la primera parte del número en hexadecimal. A continuación, cogemos el resto de dividir el número total entre doce (en nuestro ejemplo es el número 8) y hacemos la misma operación. Nos vamos a A$ y recogemos el carácter colocado en dicha posición (en nuestro ejemplo es el carácter colocado en octava posición y que es un ocho). Unimos el carácter hallado anteriormente con el que acabamos de calcular y ya tenemos el número decimal pasado a hexadecimal.

Línea 260. Definimos la variable alfanumérica A$ como un STRING con todos los dígitos del sistema de numeración hexadecimal.

Línea 270. Borramos la pantalla.

Líneas 280 y 290. Colocamos el cursor en la columna 1 de la fila 10 y preguntamos cuál es la primera dirección de memoria que queremos visualizar.

Líneas 300 y 310. Colocamos el cursor dos líneas más abajo y preguntamos cuál va a ser la última posición de memoria que queremos ver. La primera dirección la almacenamos en D1 y la última en D2.

Línea 320. Comprobamos que dichas direcciones están dentro del rango de las permitidas. En el caso de que una de ellas (o ambas no lo estén), nos volvemos a la línea **270** para volver a preguntar.

Línea 330. Comenzamos un bucle desde D1 hasta D2 con un incremento de 20, dentro del cual vamos a ver todas las posiciones de memoria entre las dos que hemos elegido. El incremento es igual a 20, porque dentro de este bucle se encuentra otro para imprimir las posiciones de memoria de 20 en 20 de la pantalla.

Línea 340. Borramos la pantalla. Cada vez que hayamos completado un grupo de 20 posiciones borramos la pantalla.

Línea 350 y 360. Imprimimos la cabecera al principio de la pantalla. Gracias a ella podremos saber qué tipo de datos son los que aparecen en la pantalla.

Línea 370. Comenzamos el segundo bucle, del que ya hemos hablado. Lógicamente este bucle va desde I hasta I + 19 para re-

correr los veinte valores que no contempla el primer bucle.

Línea 380. Hacemos que la variable PP sea igual al valor almacenado en la dirección de memoria que va marcando J. Esto lo hacemos mediante la función PEEK(n), donde n es la dirección de memoria que queremos testear.

Línea 390. Hacemos que P$ sea igual a PP, pero en hexadecimal. Para ello utilizamos la función que definimos en la línea **250.**

Línea 400. Si resulta que PP es menor que 32, entonces, como no podemos imprimir dicho carácter en la pantalla, hacemos que C$ sea igual a un asterisco.

Línea 410. En caso contrario hacemos que C$ sea igual al carácter que tiene como código ASCII a PP.

Línea 420. Imprimimos, con formato, la posición de memoria, el valor (en decimal) almacenado en ella, ese mismo valor en hexadecimal y el carácter que tiene dicho código ASCII.

Línea 430. Aquí termina el segundo bucle.

Líneas 440 y 450. Imprimimos un mensaje en la última línea de la pantalla en la cual le decimos al usuario que si quiere imprimir lo visto pulse la tecla C, y si quiere parar pulse la tecla P.

Línea 460. Leemos la tecla que ha pulsado el usuario (si la ha pulsado).

Línea 470. Si no pulsó ninguna tecla nos volvemos a la línea **460** para volver a preguntar.

Línea 480. Si el usuario pulsó la tecla C entonces nos vamos a la subrutina que empieza en la línea **530** e imprimimos lo mismo por impresora.

Línea 490. Si el usario pulsó la tecla P nos vamos a la línea **510** para terminar el programa.

Línea 500. Si pulsó una tecla distinta de las anteriores, entonces continuamos con el bucle. Aquí termina el bucle.

Línea 510. Borramos la pantalla. El programa llega a esta línea cuando se ha pulsado la P o cuando ya se han visualizado todas las posiciones de memoria deseadas.

Línea 520. Terminamos el programa.

Líneas 560 a 650. Hacemos lo mismo que hemos visto un poco más arriba, pero en vez de imprimir en la pantalla imprimimos en la impresora. Para ello sólo hemos tenido que cambiar el PRINT por LPRINT. En el COMMODORE lo que hacemos es redireccionar la salida por la impresora en vez de por la pantalla. Una vez que hemos impreso todo nos volvemos al programa principal.

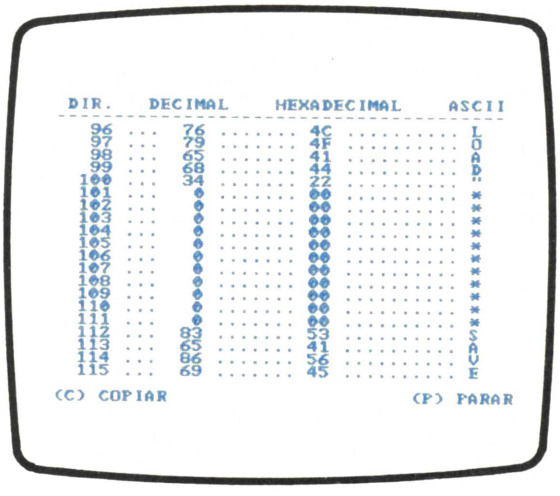


Fig. 2. Ejemplo de DUMP de memoria en el IBM.

A continuación vamos a ver cómo funciona el programa número dos en su versión para SPECTRUM.

Línea 10. Pokeamos el valor 8 en la dirección de memoria número 23658. Este poke tiene la función de hacer que todas las teclas que pulse el usuario se escriban en mayúsculas.

Línea 20. Pedimos por teclado la dirección de comienzo.

Línea 30. Comenzamos un bucle desde la dirección que ha pulsado el usuario hasta la última que tiene el ordenador (65525 = 64 Kb.) con un incremento de 21. Esto es debido a que en cada pantalla imprimiremos las direcciones en grupos de 21. Más adelante hay otro bucle que se encarga de hacerlo.

Línea 40. Borramos la pantalla. Tenemos que borrar la pantalla después de haber impreso cada grupo de direcciones.

Línea 50. Imprimimos la cabecera de la página para que el usuario sepa qué es lo que vamos a imprimir y en qué orden.

Línea 60. Comenzamos el segundo bucle que irá desde 0 hasta 20. Sumaremos el valor de la variable índice de este bucle con la del bucle anterior para saber en qué dirección de memoria tenemos que mirar.

Línea 70. Asignamos a la variable PD el valor de la dirección que nos toca mirar. Esta dirección es la suma de B más A (A + B).

Línea 80. Hacemos que P1 sea igual a PD dividido entre 16.

Línea 90. Sacamos la parte entera de P1 y la almacenamos en P. Estas dos últimas líneas tiene como propósito sacar cuál es el valor del primero de los dos dígitos que componen el número PD en hexadecimal.

Línea 100. Hacemos que LO tenga el valor de la longitud de la cadena que forma la dirección de memoria que estamos mirando.

Línea 110. Imprimimos la dirección de memoria y a continuación una serie de puntos suspensivos. El número de puntos varía según la longitud de la dirección de memoria. Así, si ésta tiene una longitud de 1, se imprimirán siete puntos. Si la longitud es 2, entonces imprimiremos 6; si es 3, imprimiremos 5, y así sucesivamente. Esto es necesario porque, como queremos que todos los datos estén alineados, cuanto más grande sea el número (más grande de longitud) menos puntos tendremos que imprimir para que todos los valores cuadren.

Línea 115. Imprimimos el valor en decimal de dicha posición de memoria. A continuación imprimimos otra serie de puntos que también varían, siguiendo la misma ley que la vez anterior. La forma de hacerlo es distinta, y se ha hecho así para que el lector pueda comprobar las dos maneras.

Línea 120. A continuación, imprimimos el primer carácter de los dos que forman el número PD en hexadecimal. Para ello miramos si P (calculado un poco más arriba) es menor que 10. En caso afirmativo lo imprimimos y ya está. Si resulta mayor de 9, entonces miramos qué valor tiene y según esté imprimimos la letra que le correnponda según el siguiente baremo:

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

Línea 130. Hacemos que P2, sea igual al resto que nos queda de dividir PD entre 16. Este número formará el segundo carácter del número PD en notación hexadecimal.

Línea 140. Asignamos a P el valor de P2.

Línea 150. Volvemos a hacer lo mismo que en la línea **120.** Esta vez lo que nos aparece es el segundo dígito en hexadecimal.

Línea 160. Por último, imprimimos el carácter que tiene como código ASCII a PD. En el caso de que dicho código ASCII sea menor de 32 (es un carácter que no se puede imprimir), o mayor de 144 (a partir de este carácter empiezan los TOLKENS), imprimiremos dos asteriscos seguidos.

Línea 170. Aquí termina el segundo bucle, con lo que también hemos impreso una pantalla entera.

Línea 180. Imprimimos en las dos líneas inferiores de la pantalla un mensaje en flash. Este avisa al usuario de que si desea copiar la pantalla por impresora tiene que pulsar la tecla C y que si quiere parar tiene que pulsar la tecla P.

Línea 190. Hacemos que el ordenador esté esperando infinitamente. Esto se consigue mediante el uso de la sentencia PAUSE 0. El ordenador estará esperando hasta que se pulse una tecla.

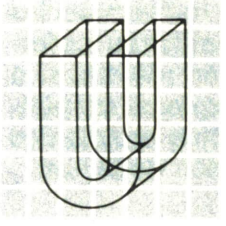
Línea 200. Una vez que se ha pulsado, la almacenamos en la variable alfanumérica K$.

Línea 210. Si la tecla pulsada es la P, entonces borramos todas las variables y la pantalla (con el uso de la sentencia CLEAR), y nos volvemos a la línea **20** para que el usuario elija otra dirección.

Línea 220. Si pulsó una C entonces copiamos la pantalla por impresora.

Línea 230. En el caso de que la tecla pulsada no fuese ni C ni P, entonces continuaríamos con el bucle para imprimir el siguiente grupo de 21 direcciones. Si hemos llegado hasta la dirección 65535 el programa se termina.

Sintetizador de voz



UNA atractiva posibilidad de comunicación hombre/máquina y que resulta muy eficiente es a través de la voz. Con dispositivos actualmente existentes se puede realizar el análisis y síntesis de la voz con calidades aceptables para ser empleados en aplicaciones reales. La síntesis de la voz humana puede realizarse de varias formas, dependiendo del método usado para su almacenamiento o interpretación:

- *Digitalización directa.* La señal captada mediante un micrófono se almacena de forma digital, después de procesarla con un conversor A/D. Permite reproducción muy fiel, pero requiere mucha capacidad de almacenamiento de las muestras. El flujo de datos necesario es del orden de 100.000 bits por segundo de voz.
- *Digitalización comprimida.* Se pierde un poco de fidelidad, pero permite reducir enormemente la memoria necesaria. Hay técnicas de codificación que permiten conservar las características principales con una gran reducción de espacio de almacenamiento. Según el sistema empleado para la compresión requiere de 10.000 a 60.000 bits por segundo de voz.
- *Simulación del sistema bucal.* La señal de voz presenta una gran redundancia, lo que permite deducir las muestras siguientes a partir de una secuencia, con una gran probabilidad. (Predicción lineal= LPC.) Mediante un modelo del sistema bucal, se simula un circuito eléctrico con parámetros constantes durante un pequeño intervalo y que se comporta como el sistema bucal para la pronunciación de cada sonido. Los parámetros cambian, según la variación de la señal a producir, cada cierto tiempo. Permite la reproducción bastante fiel con necesidades de memoria muchísimo menores que en los sistemas de digitalización comprimida. La señal producida es generalmente modulada en ancho de pulso y después filtrada. La cantidad de información resulta reducida de 1.000 a 2.000 bits por segundo.
- *Síntesis por reglas.* La producción de la señal sonora se basa en la descomposición del texto escrito en segmentos directamente sintetizables, mediante alguno de los métodos anteriores. Una forma corriente es la de encadenar alófonos, término que sirve para denominar las partes más pequeñas diferenciables de sonido producido al hablar. El número necesario de alófonos para un idioma determinado es mucho menor que el de sílabas. Podría generarse para un determinado orador, produciéndose entonces una voz, réplica de una persona determinada. Generalmente, se utiliza síntesis LPC, para la generación de los alófonos.

La señal producida por la voz tiene como características principales el tono o frecuencia fundamental y los formantes o contenido en armónicos. El tono, o 'pitch' en el argot habitual, de voz es propio de la persona que habla, siendo de baja frecuencia para voz de hombre y más alta para mujer y niño. El tono se puede modificar con relativa facilidad para producir diferentes tonalidades, manteniendo la información básica almacenada en memoria. La entonación o variación de nivel con la frase es más difícil de producir, por lo que generalmente la síntesis mediante ordenador produce una voz monótona y sin modulación, como de robot.

En el mercado existen circuitos integrados que realizan el proceso de generación a partir de señales almacenadas en forma comprimida o mediante la conversión texto-voz por síntesis mediante alófonos o elementos sonoros. Los primeros requieren tener almacenado el repertorio de palabras con las que componer las frases, mientras que los segundos componen cada palabra a partir de los elementos sonoros individuales. La señal de interrupción producida al terminar una palabra, permite encadenar de manera continua las palabras, para producir una síntesis más natural.

El sistema bucal

Interesa ver cuáles son los elementos fundamentales empleados en la producción de voz, en el sistema bucal.

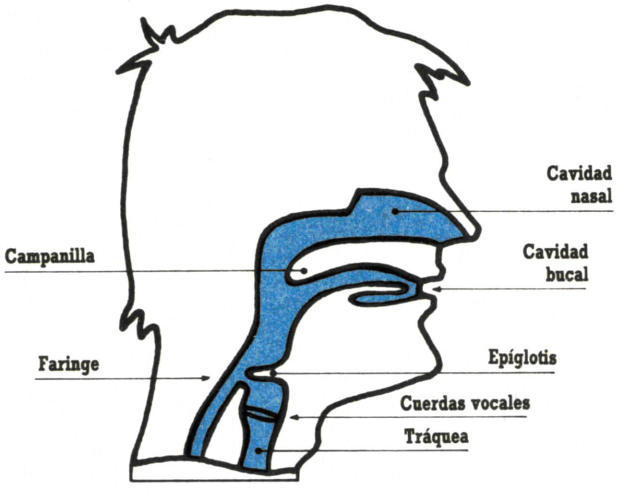


Fig. 1. Sección del sistema bucal.

Los pulmones constituyen la fuente de aire que alimenta el sistema. El diafragma es un músculo que mueve la cavidad torácica para producir la corriente de aire necesaria para la producción de sonidos. También los músculos torácicos intervienen en la respiración, pero con menor importancia para la producción de voz.

La corriente de aire producida por los pulmones origina vibraciones en las cuerdas vocales si es que está a la tensión suficiente, por el intento voluntario de hablar. La apertura o cierre rítmica, produce una vibración sonora, mientras que si es de tipo aleatorio producirá una vibración sorda.

El paso por la boca y órganos próximos, determina la modulación del sonido emitido. La cavidad bucal y las fosas nasales son los recintos que añaden sus frecuencias propias de resonancia para cambiar el timbre a la señal producida. La lengua, además, produce perturbaciones transitorias que modifican temporalmente la forma de la cavidad bucal y, por tanto, las frecuencias propias de resonancia. Estas frecuencias propias se denominan "formantes" y evolucionan suavemente a lo largo de la pronunciación de una palabra, entre las diferentes sílabas.

A cada frecuencia de vibración de las cuerdas vocales corresponde un tono característico de la persona. A cada posición de la boca corresponde un conjunto de frecuencias características de filtro, que contituyen los formantes y que también permiten identificar a cada persona. Mediante un esquema simplificado podemos ver los elementos principales que intervienen en la producción de la voz.

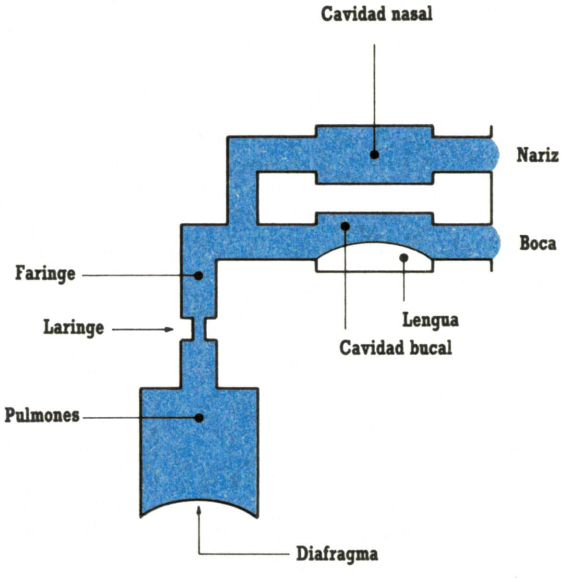


Fig. 2. Esquema simplificado del sistema bucal.

Experimentos originales

Se remontan al siglo XVIII los primeros experimentos para producir sonidos similares a la voz humana. Muchos órganos de tubos disponían de un registro de trompetas denominado "voz de viejo". No era realmente una síntesis como ahora la entendemos, pero daba la sensación de ser producida por voz humana.

Mediante cavidades resonante, alimentadas por un flujo constante de aire y controlado por lengüetas, un teclado similar al de un piano permitía la activación a voluntad de sonidos similares a la voz humana.

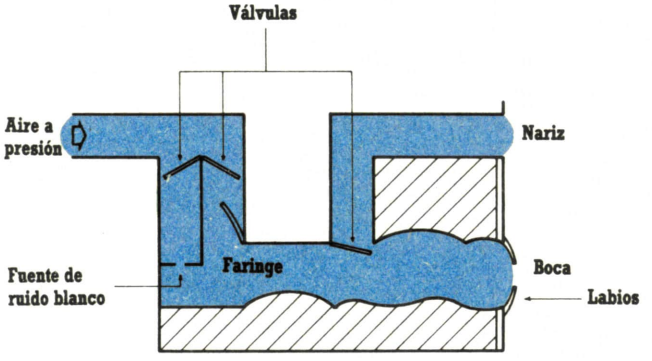


Fig. 3. Sistema mecánico de síntesis.

Un experimento mucho más en línea con la síntesis electrónica, basaba su funcionamiento en el control mecánico de un símil del sistema bucal. Las válvulas son accionadas mediante un teclado y el flujo de aire se controla con pedales. Un operador entrenado podía producir sonidos parecidos a voz.

Pero el primer sistema digno de mención por su similaridad con la producción real de sonido es el VOCODER, que mediante el análisis de la voz permite conocer qué es lo que hay que controlar en la producción de un sonido cualquiera.

Sistemas reales

Una forma de producir sonido similar a la voz humana de forma electrónica consiste en reproducir analógicamente cada elemento

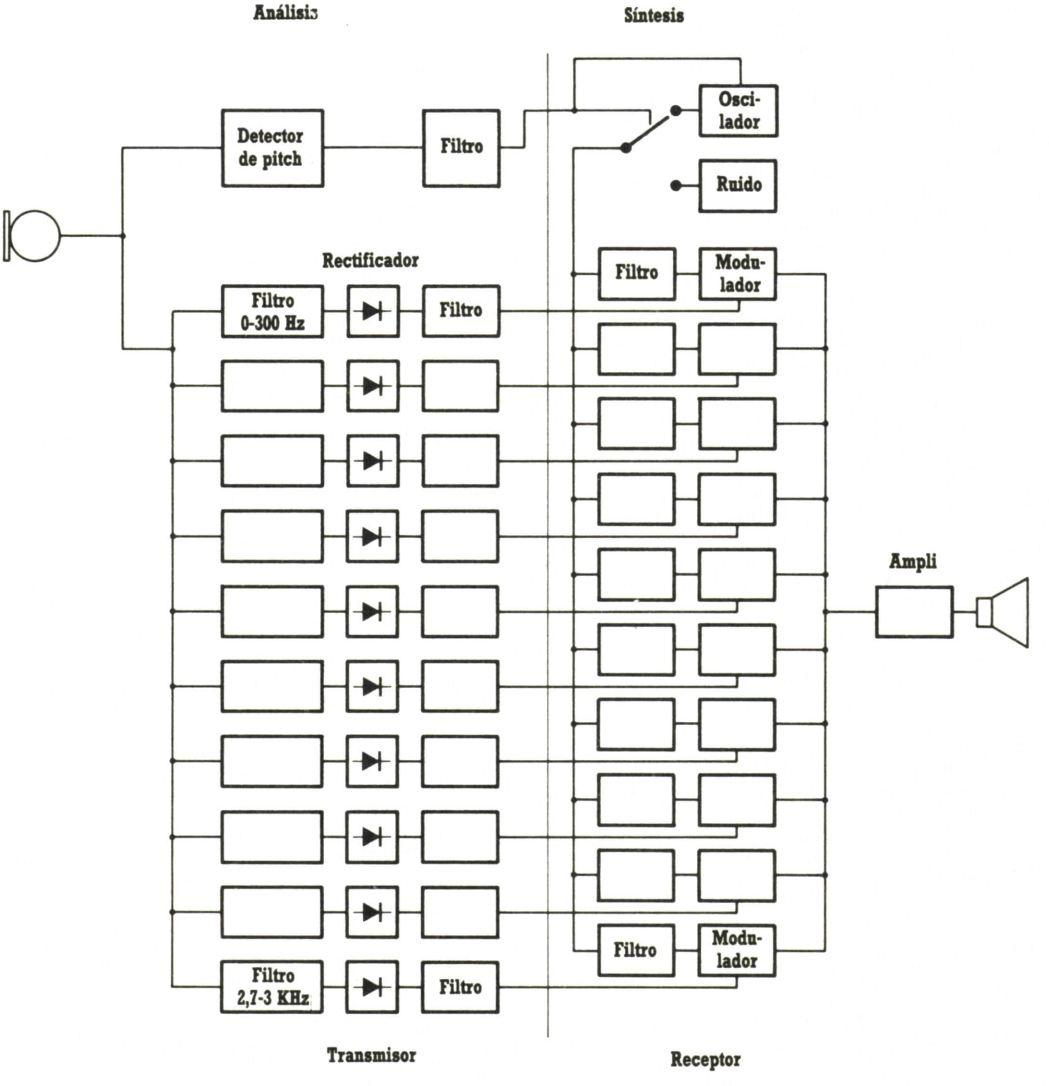


Fig. 4. Esquema simplificado de VOCODER.

de los descritos en los bloques y controlarlo mediante un ordenador.

Las fuentes de sonido han de ser dos: una señal periódica que identifica el tono o pitch y otra de ruido aleatorio o blanco, para la producción de las señales sordas. Además, el sistema fonador se representará por un filtro, compuesto por resonadores, cuyas frecuencias propias pueden variarse. El control periódico de todos los parámetros de control dará origen a la producción de una secuencia de sonidos sintéticos, similares a los propios de una persona determinada. Hemos de identificar para cada sonido necesario la combinación de parámetros y su duración, para la producción apropiada y su relación con el sonido que precede o sigue. Las cantidades mínimas de sonido que requieren unos parámetros constantes en el sistema se denominan alófonos. Estos son combinaciones de sonidos vocálicos y consonantes que resultan indivisibles, para una combinación vocal-consonante o solamente vocal.

Aplicación práctica

En el esquema de la figura se muestran los bloques necesarios para la síntesis de voz mediante alófonos, a partir de una tabla de palabras almacenada en una memoria interna del sintetizador SP0256A-AL2. Existe una variante del circuito que dispone de conexión para una ROM externa de alófonos, que permite experimentar con la síntesis de otros idiomas, aunque la generación de una nueva memoria no es tarea sencilla.

Los bloques necesarios para la síntesis de voz mediante alófonos son:

- Interfaz con el ordenador. Utilizar el circuito de puerto de salida para la programación del código del alófono a producir. A un puerto de entrada se llevará la señal de indicación de conversión en proceso.
- Circuito sintetizador SP0256A-AL2, de General Instruments.
- Memoria de palabras, si se emplea la versión de ROM externa.
- Amplificador, filtro y control de volumen.
- Altavoz o auricular.

El circuito puede montarse como extensión de la tarjeta de ampliación de puertos de entrada salida, pues solamente requiere uno de cada tipo. La señal generada por el circuito de síntesis es de tipo modulación por ancho de pulso, que necesita ser filtrada y amplificada para ser reproducida por un altavoz.

Programa

La producción de una secuencia de alófonos para generar una palabra consiste en la presentación en el puerto de salida del código del sonido a producir indicando con un pulso (ALD) en el momento en que está disponible. Para encadenar el siguiente sonido deberá consultarse la señal LRQ o SBY para sa-

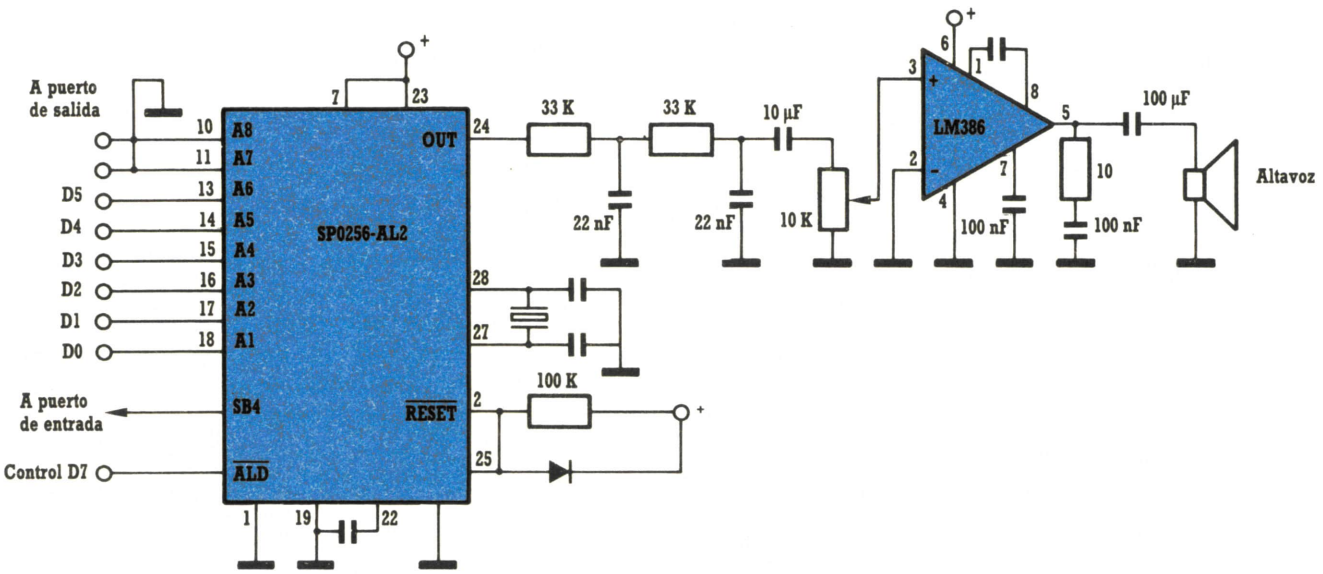


Fig. 5. Sintetizador de voz.

ber cuándo el registro de código está libre. Los códigos posibles incluyen todos los sonidos propios de un idioma y varios períodos de silencio. En la tabla se indican los códigos disponibles.

La tabla indica que solamente hay disponibles 64 alófonos en la ROM interna del circuito sintetizador.

Para generar un secuencia de sonidos correspondientes a un texto es necesario descomponer cada palabra en los alófonos más próximos a los disponibles en la tabla, teniendo en cuenta las reglas de formación de cada lenguaje. El problema principal es tener en cuenta las excepciones para poder proporcionar un sonido natural.

El programa de ejemplo muestra la activación del prototipo utilizando lenguaje BASIC y secuencias preprogramadas para producir algunas palabras.

El circuito mostrado es un dispositivo apropiado para experimentar, aunque la generación de voz natural requeriría un programa más elaborado.

Tabla de alófonos para idioma inglés en SP0256A-AL2

Código HEX	Alófono	Ejemplo	Duración mS	Código HEX	Alófono	Ejemplo	Duración mS
00	PA1	Pausa	10	20	AW	Out	370
01	PA2	Pausa	30	21	DD2	Do	160
02	PA3	Pausa	50	22	GG3	Wig	140
03	PA4	Pausa	100	23	VV	Vest	190
04	PA5	Pausa	200	24	GG1	Got	80
05	OY	Boy	420	25	SH	Ship	160
06	AY	Sky	260	26	ZH	Azure	190
07	EH	End	70	27	RR2	Brain	120
08	KK3	Comb	120	28	FF	Food	150
09	PP	Pow	210	29	KK2	Sky	190
0A	JH	Dodge	140	2A	KK1	Can't	160
0B	NN1	Thin	140	2B	ZZ	Zoo	210
0C	IH	Sit	70	2C	NG	Anchor	220
0D	TT2	To	140	2D	LL	Lake	110
0E	RR1	Rural	170	2E	WW	Wool	180
0F	AX	Succeed	70	2F	XR	Repair	360
10	MM	Milk	180	30	WH	Whig	200
11	TT1	Part	100	31	YY1	Yes	130
12	DH1	They	290	32	CH	Church	190
13	IY	See	250	33	ER1	Fir	160
14	EY	Beige	280	34	ER2	Fir	300
15	DD1	Could	70	35	OW	Beau	240
16	UW1	To	100	36	DH2	They	240
17	AO	Aught	100	37	SS	Vest	90
18	AA	Hot	100	38	NN2	No	190
19	YY2	Yes	180	39	HHC	Hoe	180
1A	AE	Hat	120	3A	OR	Store	330
1B	HH1	He	130	3B	AR	Alarm	290
1C	BB1	Business	80	3C	YR	Clear	350
1D	TH	Thin	180	3D	GG2	Guest	40
1E	UH	Book	100	3E	EL	Saddle	190
1F	UW2	Food	260	3F	BB2	Business	50

```
10 REM  VOZPC
20 REM SINTESIS DE VOZ UTILIZANDO SP0256A-AL2
30 REM PARA TARJETA SOBRE IBM-PC
40 DIM A(50)
42 REM DIRECCIONES DE PUERTOS
45 ENTRADA=&H201: SALIDA=&H201
50 REM  LEE CODIGOS PARA LOS ALOFONOS
60 FOR I=1 TO 7
70     READ A(I)
80 NEXT I
90 REM   PRONUNCIA HELLO BOY
100 FOR I=1 TO 7
110    M=INP(ENTRADA)
120    IF M=0 THEN 110
130    OUT SALIDA,A(I)
140 NEXT I
150 REM ADMITE SECUENCIA DE ALOFONOS
160 REM Y LA PRONUNCIA
170 PRINT "TECLEE LOS NUMEROS DE ALOFONO UNO POR UNO"
180 I=1
190 WHILE (X<>64 AND I<50)
200     INPUT "NUMERO DE ALOFONO O 64 PARA FIN";X
210     IF X<0 OR X>64 THEN PRINT "NUMERO INCORRECTO": GOTO 200
220     A(I)=X:I=I+1
230 WEND
240 NMAX=I-2
250 FOR I=1 TO NMAX
260    M=INP(ENTRADA)
270    IF M=0 THEN 260
280    OUT SALIDA,A(I)
290    PRINT A(I);" ";
300 NEXT I
310 END
320 DATA 57,45,53,2,28,5,4
```

Conclusión

La síntesis de voz es una aplicación fácilmente asequible para cualquier ordenador personal, utilizando alguno de los circuitos como el indicado, que realizan todas las operaciones de generación de la señal acústica. Otros circuitos similares son:

- TMS0280, TMS5110 y TMS5220 de Texas Instruments, de vocabulario limitado.
- SPC-MM52164 y SPC-MM54104 de NationalSemiconductors, también denominado Digitalker, para número limitado de palabras sintetizadas con gran calidad.
- SC-01 de Votrax que utiliza síntesis por fonemas.
- SSI-263 de Silicon Systems también de síntesis por fonemas.

Actualmente se puede disponer de muchos equipos de bajo coste que utilizan alguno de estos circuitos para sintetizar voz, con la que comunicar con el usuario, en aplicaciones tan simples como termómetro, balanzas, ayudas a la enseñanza o alarmas en el automóvil.

El análisis de voz es un proceso más complejo y en la actualidad sólo puede hacerse en ordenador personal el reconocimiento

de palabras sueltas o frases cortas, quedando el problema de reconocimiento continuo de voz para máquinas muy grandes. La posibilidad de emplear los procesadores de señal de grandes prestaciones ha permitido realizar aplicaciones de reconocimiento y síntesis de voz en entornos restringidos, como el diálogo con el ordenador, para indicarle acciones a ejecutar.

El tratamiento de voz va a ser un medio importante para hacer accesibles las posibilidades de la informática a colectivos que en este momento no pueden servirse de ella por alguna incapacidad física, visual o manual.

NATURALEZA Y TECNOLOGIA

Estructura y composición de La Tierra

PODEMOS considerar nuestro planeta como un gran globo formado de diversos materiales que tienen densidad diferente, presentan distintos estados físicos y están situados formando capas concéntricas.

Nosotros vivimos en la parte sólida a la que llamamos corteza terrestre o litosfera. Pero esta corteza está cubierta en sus tres cuartas partes por las aguas que forman los océanos y mares. Esta masa de agua constituye la hidrosfera.

Rodeando la litosfera y la hidrosfera hay una capa de sustancia gaseosa que recibe el nombre de atmósfera.

La endosfera, capa interior a la corteza terrestre, está compuesta por materiales de densidad muy elevada y cuya composición se desconoce.

La atmósfera consta de las siguientes capas:

- Troposfera
- Estratosfera
- Ionosfera

La litosfera se divide en sima y sial.

Por último, en la endosfera se distin-

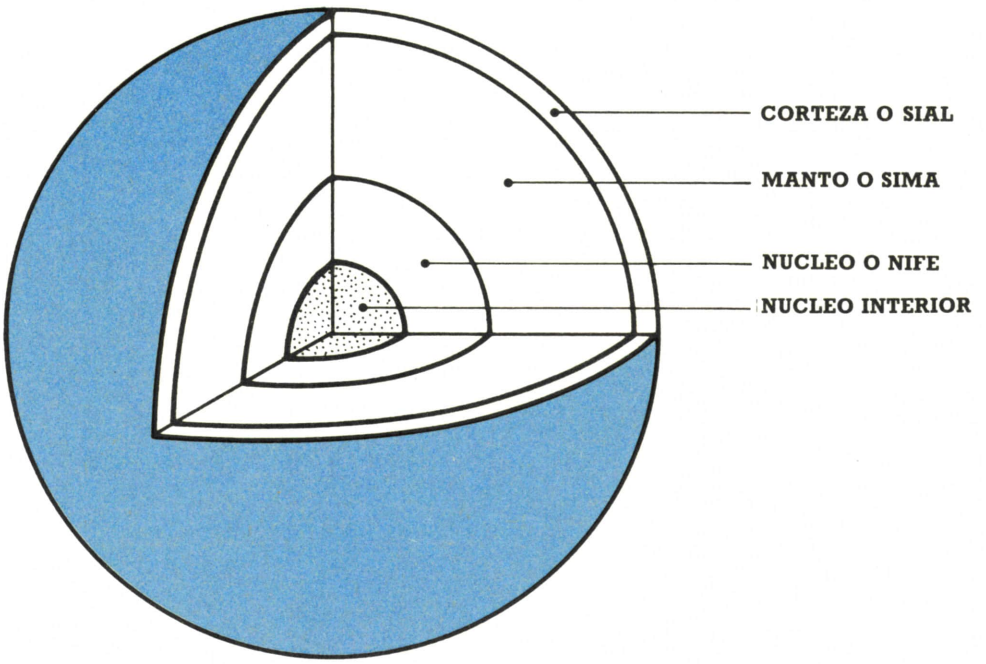


Fig. 1. Estructura y composición de La Tierra.

guen tres zonas concéntricas: manto, zonas intermedias y núcleo o nife.

Sobre el programa

El programa consta de dos partes: la primera te servirá para memorizar la disposición física de las capas de nuestro planeta, ayudado por su representación gráfica; en la segunda, comprobarás tus conocimientos sobre este tema.

Si tu ordenador es IBM o MSX puedes hacer que el dibujo sea aún más atractivo introduciendo colores de tinta. En la línea 120, por ejemplo, sería circle(128,80),10,11,0,1.57

Repite el color 11 en las líneas comprendidas entre la 60 y la 190. Te aconsejamos emplear el color 15 en las lineas 293, 300 y 310.

```
10 REM ============================
20 REM  ESTRUCTURA DE LA TIERRA
30 REM ============================
40 CLS
50 SCREEN 1
51 PRINT:PRINTTAB(2)"ESTRUCTURA Y COMPOSICION DE LA TIERRA"
52 REM DIBUJO DE LA TIERRA
60 CIRCLE (128,80),60
70 CIRCLE (128,80),57,4.71,1.57
80 CIRCLE (128,80),30,4.71,1.57
90 CIRCLE (128,80),60,1.57,4.71,2
100 CIRCLE (128,80),56,1.57,4.71,2
110 CIRCLE (128,80),30,1.57,4.71,2
120 CIRCLE (128,80),10,0,1.57
130 CIRCLE (128,80),10,4.2,0,1/2
140 CIRCLE (128,80),10,1.6,3.7,2
150 CIRCLE (128,80),30,4.3,0,1/2
160 LINE (128,70)-(128,23)
170 LINE (128,110)-(128,137)
180 LINE (138,82)-(158,82)
190 LINE (122,85)-(116,92)
200 LOCATE 195,75:PRINT "manto"
210 LOCATE 200,83:PRINT "o"
220 LOCATE 195,91:PRINT "sima"
230 LINE (170,80)-(193,80)
240 LOCATE 180,150:PRINT "nucleo"
250 LOCATE 185,158:PRINT "o"
260 LOCATE 180,166:PRINT "nife"
270 LOCATE 195,25:PRINT "corteza"
280 LOCATE 198,33:PRINT "o"
290 LOCATE 195,41:PRINT "sial"
291 LOCATE 50,150:PRINT "nucleo"
292 LOCATE 50,158:PRINT "interior"
293 LINE (128,80)-(60,145)
300 LINE (168,37)-(193,30)
310 LINE (147,97)-(185,147)
320 FOR J=1 TO 4000
321 NEXT J
322 SCREEN 0
330 DIM P$(10,2),R$(10,3),S(10)
339 REM ====================
340 REM LECTURA DE LOS DATOS
350 REM ====================
360 FOR I=1 TO 10
370 FOR K=1 TO 2
380 READ P$(I,K)
390 NEXT K
400 FOR J=1 TO 3
410 READ R$(I,J)
420 NEXT J
430 READ S(I)
440 NEXT I
450 REM ================
460 REM TEST DE PREGUNTAS
470 REM ================
480 CLS
490 LET C=0
500 FOR I=1 TO 10
510 PRINT:PRINT:PRINTTAB(5)"PREGUNTA";I
520 PRINTTAB(5)"--------":PRINT:PRINT
530 FOR K=1 TO 2
540 PRINTTAB(3) P$(I,K)
550 NEXT K:PRINT:PRINT
560 FOR J=1 TO 3
570 PRINT J;R$(I,J):PRINT
580 NEXT J
```

```
590 PRINT:PRINT:PRINT:PRINTTAB(3)"ELIGE LA CORRECTA"
600 INPUT SN$
601 IF SN$<>"1" AND SN$<>"2" AND SN$<>"3"THEN CLS:PRINT:PRINT:PRINT:PRINTTAB(5)"
 CONTESTA 1 2 O 3":FOR K=1 TO 600:NEXT K:CLS:GOTO 510
610 IF S(I)=VAL(SN$)THEN GOSUB 800
611 IF S(I)<>VAL(SN$) THEN GOSUB 910
620 CLS
630 NEXT I
640 PRINT:PRINT:PRINT:PRINTTAB(5) "HAS TENIDO ";C;"RESPUESTAS CORRECTAS"
650 FOR K=1 TO 1000
660 NEXT K
670 CLS
680 IF C>=5 THEN PRINT:PRINT:PRINT:PRINT:PRINTTAB(10)"ENHORABUENA"
681 IF C<5 THEN PRINT:PRINT:PRINT:PRINTTAB(10)"ESTUDIATELO OTRA VEZ"
690 DATA "COMO SE DENOMINA LA PARTE","SOLIDA DE LA SUPERFICIE TERRESTRE","HIDROS
FERA","LITOSFERA","ATMOSFERA",2
700 DATA "COMO SE DENOMINA LA PARTE","INTERIOR DE LA TIERRA","LITOSFERA","ENDOSF
ERA","HIDROSFERA",2
710 DATA "DI CUAL DE LAS SIGUIENTES","CAPAS NO PERTENECEN A LA ENDOSFERA","MANTO
 ","NUCLEO","SIAL",3
720 DATA "CUALES SON LOS COMPONENTES","QUIMICOS PRINCIPALES DEL SIAL","SILICE Y
MAGNESIO","SILICE Y ALUMINIO","HIERRO Y SILICE",2
730 DATA "COMO SE LLAMA LA ZONA DE TRANSICION","SITUADA A CONTINUACION DEL SIAL"
,"DISCONTINUIDAD DE MOHOROVIC","DISCONTINUIDAD DE CONRAD","DISCONTINUIDAD DE GUT
EMBERG",1
740 DATA "CUAL SE SUPONE QUE ES"," EL ESTADO DEL NUCLEO","SOLIDO","LIQUIDO","GAS
EOSO",2
750 DATA "COMO SE LLAMA LA PARTE GASEOSA"," QUE RODEA A LA TIERRA","LITOSFERA","
CORTEZA","ATMOSFERA",3
760 DATA "CUAL DE LAS SIGUIENTES CAPAS","PERTENECE A LA ATMOSFERA","HIDROSFERA",
"ENDOSFERA","TROPOSFERA",3
770 DATA "A QUE CAPA PERTENECEN"," LOS CONTINENTES Y CUENCAS OCEANICAS","MANTO",
"NUCLEO","CORTEZA",3
780 DATA "LA PLATAFORMA CONTINENTAL Y","LAS LLANURAS ABISALES SON PARTE DE...","
LOS CONTINENTES","LAS CUENCAS OCEANICAS","SIMA",2
790 END
800 REM ==========
810 REM  CORRECTA
820 REM ==========
830 CLS
840 FOR F=1 TO 9 :PRINT:NEXT F:PRINTTAB(10)"^^^^^^^^^"
850 PRINTTAB(10) "CORRECTA"
860 PRINTTAB(10) "--------"
870 FOR K=1 TO 600
880 NEXT K
890 LET C=C+1
900 RETURN
910 REM ============
920 REM  INCORRECTA
930 REM ============
940 CLS
950 FOR F=1 TO 9:PRINT:NEXT F:PRINTTAB(10)" NO ES CORRECTA"
960 PRINT:PRINT:PRINT:PRINTTAB(2)"LA RESPUESTA CORRECTA ES ":PRINT:PRINT:PRINT:P
RINTTAB(5) R$(I,S(I))
970 FOR K=1 TO 1500
980 NEXT K
990 RETURN
```

Modificaciones para otros equipos

AMSTRAD

```
50 no se pone
60 origin 110,200:deg
70 let r1=110 : let r2=90 : let r3=30
80 for i=1 to 360
90 plot r1*cos(i),r1*sin(i)
100 plot r2*cos(i),r2*sin(i)
110 plot r3*cos(i),r3*sin(i)
120 next i
130 plot 30,100: draw 250,100: locate 27,7:print "Corteza"
```

```
140 plot 20,60:draw 250,60: locate 27,9: print "Manto"
150 plot 0,0:draw 250,0: locate 27,13: print "Nucleo"
Suprime las lineas desde la 160 hasta la 310
322 no se pone
```

SPECTRUM

```
50 no se pone
60 circle 46,88,47
70 circle 46,88,39
80 circle 46,88,13
100 plot 53,132 :draw 93,0
110 plot 54,114: draw 100,0
120 plot 46,88: draw 106,0
130 print at 7,27;"Corteza"
140 print at 9,27;"Manto"
150 print at 13,27;"Núcleo"
790 goto 9999
suprime las lineas desde la 160 hasta la 310
```

IBM

```
50 screen 2
200 locate 75,195
210 locate 83,200
220 locate 91,195
240 locate 150,180
250 locate 158,185
260 locate 166,180
270 locate 25,195
280 locate 33,198
290 locate 41,195
291 locate 150,50
292 locate 158,50
```

COMMODORE

Suprime las lineas desde la 50 hasta la 310

MATEMATICAS

Mínima distancia entre dos rectas que se cruzan

Las ecuaciones vectoriales de dos rectas en el espacio son:

r: $x=a+tv$ r': $x=b+sw$

Donde r pasa por el punto A (a_1,a_2,a_3) y tiene la dirección del vector v (v_1,v_2,v_3) y r' pasa por el punto B (b_1,b_2,b_3) y tiene la dirección de w (w_1,w_2,w_3); t y s son los parámetros variables de la ecuación.

El proceso que seguimos para calcular la distancia mínima entre estas dos rectas es el siguiente.

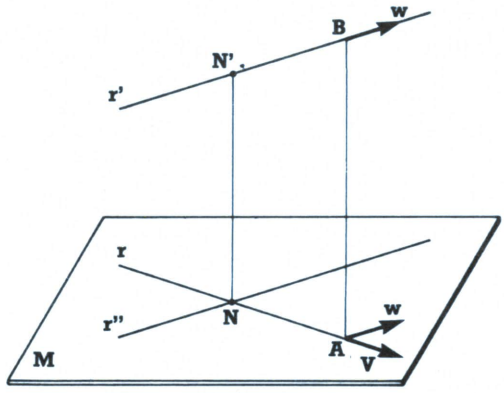
Trazamos el plano M que pasa por r y es paralelo a r'. Este plano pasará por A y será paralelo a v y w.

La ecuación del plano M se obtiene de:

$$\begin{vmatrix} v_1 & w_1 & X-a_1 \\ v_2 & w_2 & Y-a_2 \\ v_3 & w_3 & Z-a_3 \end{vmatrix} = \emptyset$$

Todos los puntos de r' distan lo mismo del plano M; esa distancia es la distancia mínima y su valor numérico será:

$$DIST = \frac{\begin{vmatrix} v_1 & w_1 & b_1-a_1 \\ v_2 & w_2 & b_2-a_2 \\ v_3 & w_3 & b_3-a_3 \end{vmatrix}}{\sqrt{\begin{vmatrix} v_2 & w_2 \\ v_3 & w_3 \end{vmatrix}^2 + \begin{vmatrix} v_1 & w_1 \\ v_3 & w_3 \end{vmatrix}^2 + \begin{vmatrix} v_1 & w_1 \\ v_2 & w_2 \end{vmatrix}^2}}$$



Sobre el programa

Para calcular la distancia mínima, el ordenador te pedirá las coordenadas de cuatro puntos: dos pertenecientes a la recta r, y, los otros dos, pertenecientes a la recta r'.

Con estos puntos obtiene las ecuaciones paramétricas de cada recta, que te presentará en pantalla.

A continuación, realiza, paso a paso, el siguiente proceso:

1. Trazar las dos rectas.
2. Trazar el plano que pasa por r
3. Cálculo de la distancia de un punto de r' a ese plano.

Las líneas 420, 610, 650 controlan el tiempo que aparece el dibujo en la pantalla. Puedes modificarlas si lo consideras oportuno.

```
10 REM ==================================================
20 REM  MINIMA DISTANCIA ENTRE DOS RECTAS QUE SE CRUZAN
30 REM ==================================================
40 CLS
50 PRINT:PRINT
60 PRINT:PRINT
70 PRINTTAB(2)"introduce un punto de r"
80 INPUT"(coordenada X)";RA
90 INPUT "(coordenada Y)";RB
100 INPUT "(coordenada Z)";RC
110 PRINT
120 PRINTTAB(2)"introduce otro punto de r"
130 INPUT "(coordenada X)";RRA
140 INPUT "(coordenada Y)";RRB
150 INPUT "(coordenada Z)";RRC
160 CLS
170 PRINT:PRINT:PRINT
180 PRINTTAB(2) "introduce un punto de r'"
190 INPUT "(coordenada X)";SA
200 INPUT "(coordenada Y)";SB
210 INPUT "(coordenada Z)";SC
220 PRINT
230 PRINTTAB(2)"introduce otro punto de r'"
240 INPUT "(coordenada X)";SSA
250 INPUT "(coordenada y)";SSB
260 INPUT "(coordenada Z)";SSC
270 LET VA=RRA-RA
280 LET VB=RRB-RB
290 LET VC=RRC-RC
300 LET WA=SSA-SA
310 LET WB=SSB-SB
320 LET WC=SSC-SC
330 CLS
340 PRINT:PRINT
350 PRINT:PRINT
360 PRINTTAB(2)"LAS ECUACIONES PARAMETRICAS SON:"
370 PRINT:PRINT
380 PRINT
390 PRINTTAB(2)  "X=";RA;"+t";VA;"      X=";SA;"+t";WA
400 PRINTTAB(2)  "Y=";RB;"+t";VB;"      Y=";SB;"+t";WB
410 PRINTTAB(2)  "Z=";RC;"+t";VC;"      Z=";SC;"+t";WC
420 FOR I=1 TO 2000
430 NEXT I
```

```
440 REM ====================
450 REM DIBUJO DE LAS RECTAS
460 REM ====================
470 SCREEN 1
480 PRINTTAB(3) "1.-Trazamos las dos rectas"
490 LINE (50,70)-(170,70)
500 LINE (60,120)-(110,90)
510 FOR I=1 TO 600
520 NEXT I
530 PRINTTAB(3) "2.-Trazamos el plano que pasa por r"
540 REM =================
550 REM DIBUJO DEL PLANO
560 REM =================
570 LINE (55,80)-(175,80)
580 LINE (40,150)-(160,150)
590 LINE (55,80)-(40,150)
600 LINE (160,150)-(175,80)
610 FOR I=1 TO 1000
620 NEXT I
630 PRINTTAB(3)"3.-Hallamos la distancia de un punto de r' al plano"
640 LINE (87,105)-(87,70)
650 FOR I=1 TO 600
660 NEXT I
670 LET T=1
680 LET BA=SA+T*WA
690 LET BB=SB+T*WB
700 LET BC=SC+T*WC
710 LET AA=RA+T*VA
720 LET AB=RB+T*VB
730 LET AC=RC+T*VC
740 LET A=BB-AB
750 LET B=BC-AC
760 DEF FNSOLDET(X1,X2,Y1,Y2)=X1*Y2-X2*Y1
770 LET A=SB-RB
780 LET B=SC-RC
790 LET NUM=VA*FNSOLDET(WB,A,WC,B)-  WA*FNSOLDET(VB,A,VC,B)+(BA-AA)*FNSOLDET(VB,
WB,VC,WC)
800 REM ===============
810 REM COMPROBACIONES
820 REM ===============
830 IF NUM = 0 THEN CLS: PRINTTAB(5)"NO SE CRUZAN"
840 FOR H=1 TO 1000
850 NEXT H
860 CLS
870 REM ============================
880 REM  VISUALIZACION DE RESULTADOS
890 REM ============================
900 LET U=FNSOLDET(VB,WB,VC,WC)
910 LET O=FNSOLDET(VA,WA,VC,WC)
920 LET J=FNSOLDET(VA,WA,VB,WB)
930 LET DEN=(U^2+O^2+J^2)^.5
940 LET DIST=ABS(NUM/DEN)
950 PRINT:PRINT
960 PRINT:PRINT
970 PRINTTAB (3) "LA DISTANCIA MINIMA ES:";DIST
980 FOR H=1 TO 3000
990 NEXT H
```

Modificaciones para otros equipos

COMMODORE

```
40 print chr$(147)
160 print chr$(147)
330 print chr$(147)
470 no se pone
830 if num=0 then print chr$(147):print tab(5) "No se cruzan"
860 print chr$(147)
```

AMSTRAD

```
470 no se pone
490 plot 50,70 :drawr 120,0
500 plot 60,120 :drawr 50,-30
570 plot 55,80 : drawr 120,0
580 plot 40,150 : drawr 120,0
590 plot 55,80 : drawr -15,70
600 plot 160,150 : drawr 15,-70
```

IBM

```
470 screen 2
```

SPECTRUM

```
470 no se pone
490 plot 50,70 :draw 120,0
500 plot 60,120 :draw 50,-30
570 plot 55,80 : draw 120,0
580 plot 40,150 : draw 120,0
590 plot 55,80 : draw -15,70
600 plot 160,150 : draw 15,-70
```

SOCIEDAD

Las civilizaciones precolombinas

El descubrimiento del Nuevo Mundo supone, además, el encuentro entre distintas culturas.

Los conquistadores encontrarán pueblos que han quedado detenidos en la Edad de Piedra, pero también grandes civilizaciones: aztecas, mayas e incas.

Los aztecas son originarios del norte de Méjico. Poseen una escritura compuesta de signos simbólicos y notables conocimientos de astronomía, botánica y medicina. Su capital es la ciudad de Tenochtitlán, edificada sobre las islas del lago Texcoco.

Los mayas asombran por sus conocimientos de astronomía, matemáticas y sobre el calendario. En efecto, si bien la preocupación por el calendario puede encontrarse en muchas culturas, es en la maya donde llega a una mayor perfección.

Los incas, en América del Sur, sobresalen por su arquitectura, su metalurgia con soldadura, por su red de caminos, por la unificación lingüística con el quechúa, que se convierte en lengua oficial del imperio y por su concepción del Estado. Es ésta su nota más característica. El Estado tenía una estructura piramidal, cuyo vértice lo ocupaba el Inca, hijo del Sol y jefe político, religioso y militar a la vez. Dotados de un gran sentido de la organización y la disciplina, crearon el mayor imperio y la más notable estructura política de la América precolombina.

El programa

El programa pone a prueba tus conocimientos sobre estas culturas.

Consta de 10 preguntas tipo test, cada una con tres posibles respuestas entre las que deberás seleccionar la correcta.

Cuando hayas contestado todas la preguntas el ordenador te indicará el número de aciertos y fallos que has tenido.

```
10 REM================================
20 REM CIVILIZACIONES PRECOLOMBINAS
30 REM================================
40 DIM PR$(10,2),RP$(10,4),SO$(10)
50 REM================================
60 REM   LECTURA
70 REM================================
```

```
80 FOR I=1 TO 10
90 FOR K=1 TO 2
100 READ PR$(I,K)
110 NEXT K
120 FOR J=1 TO 4
130 READ RP$(I,J)
140 NEXT J,I
150 REM================================
160 REM VISUALIZACION POR PANTALLA
170 REM================================
180 CLS
190 FOR F= 1 TO 9:PRINT:NEXT  F:PRINTTAB(10)"TEST SOBRE "
200 FOR F=1 TO 7: PRINT :NEXT F:PRINTTAB(3)    " LAS CIVILIZACIONES PRECOLOMBINA
S "
210 PRINT:PRINTTAB(3)  " === ============== ============="
220 FOR H=1 TO 2000
230 NEXT H:CLS
240 LET N=0
250 FOR I=1 TO 10
260 PRINT :PRINT:PRINTTAB(5)  "PREGUNTA NUMERO";I
270 PRINTTAB(5) "======== ======":PRINT:PRINT
280 FOR K=1 TO 2
290 PRINTTAB(7) PR$(I,K)
300 NEXT K:PRINT:PRINT
310 FOR J=1 TO 3
320 PRINT J;RP$(I,J):PRINT
340 NEXT J
350 PRINT:PRINT
370 PRINT "SELECCIONA LA CORRECTA";
380 INPUT SO$(I)
390 LET Y$=SO$(I)
400 IF Y$<>"1" AND Y$<>"2" AND Y$<>"3" THEN CLS:PRINT:PRINT:PRINT:PRINT:PRINTTAB
(5) "ESCOGE LA OPCION 1 2 O 3":FOR K=1 TO 1000:NEXT K:CLS:GOTO 260:
410 IF SO$(I)=RP$(I,4)THEN GOSUB 620
411 IF SO$(I)<>RP$(I,4) THEN GOSUB 730
420 CLS
430 NEXT I
440 FOR F=1 TO 10:PRINT:NEXT F:PRINTTAB(5) "HAS ACERTADO  ";N;" RESPUESTAS"
460 FOR K=1 TO 1000
470 NEXT K
480 CLS
490 IF N>=5 THEN PRINT:PRINT:PRINT:PRINT:PRINTAB(10)"ENHORABUENA"
500 IF N<5 THEN PRINT:PRINT:PRINT:PRINT:PRINTAB(10) "ESTUDIA MAS"
510 DATA" DONDE SE SITUABA"," LA CULTURA INCAICA?","DESDE ECUADOR A CHILE","MEXI
CO","PENINSULA DEL YUCATAN","1"
520 DATA"QUIENES CONSTRUYERON"," LA PIRAMIDE DEL SOL?","AZTECAS","MAYAS","INCAS"
,"1"
530 DATA"DE QUE IMPERIO ERA CAPITAL","    TENOCHTILTAN?","MAYA","AZTECA","INCA",
"2"
540 DATA"QUIEN FUE EL FUNDADOR","  DEL IMPERIO INCA?","INCA YUPANQUI","TOPA INCA
","HUAYNA CAPAC","1"
550 DATA"QUIEN SE APODERO"," DEL IMPERIO INCAICO?","PIZARRO","COLON","HERNAN COR
TES","1"
560 DATA"QUE CULTURA SE SITUABA","EN EL ACTUAL MEXICO?","AZTECA","MAYA","INCA","
1"
570 DATA"DE QUE IMPERIO ERA EMPERADOR","     MOCTEZUMA?","AZTECA","MAYA","INCA",
"1"
580 DATA"QUIEN ERA EL PRINCIPAL DIOS"," DE LOS AZTECAS?","CHAC","HUITZILOPOCHTLI
","EL SOL","2"
590 DATA"QUIENES INVENTARON UN SISTEMA"," DE ESCRITURA JEROGLIFICA?","MAYAS","AZ
TECAS","INCAS","1"
600 DATA"QUE RELIGION SE BASABA"," EN EL CULTO AL SOL?","AZTECA","MAYA","INCA","
3"
610 END
620 REM================================
630 REM RESPUESTA CORRECTA
640 REM================================Ok
650 CLS
660 FOR F=1 TO 9:PRINT:NEXT F:PRINTTAB(10)  "========"
670 PRINTTAB(10) "CORRECTO"
680 PRINTTAB(10) "========"
690 FOR K=1 TO 600
700 NEXT K
710 LET N=N+1
720 RETURN
730 REM=====================
740 REM RESPUESTA ERRONEA
750 REM=====================
760 CLS
770 FOR F=1 TO 10:PRINT:NEXT F:PRINTTAB(5) "NO ES CORRECTO"
```

```
780 LET X=VAL(RP$(I,4))
790 PRINT:PRINT:PRINT:PRINTTAB(2) "LA RESPUESTA CORRECTA ES ":PRINT:PRINT:PRINT:
PRINTTAB(5) RP$(I,X)
800 FOR K=1 TO 900
810 NEXT K
820 RETURN
```

Modificaciones para otros equipos

El programa es válido para Amstrad ,IBM y MSX

Si tienes un Commodore sustituye la palabra CLS por la sentencia PRINT CHR$(147) en las líneas 180 ,400 ,420 ,480 ,650 y 760

En el Spectrum utiliza nombres de variables con subíndice de una sola letra y cambia la línea 610 por goto 9999

PARA LOS MAS JOVENES

Aprendiendo a dividir

Este programa te servirá para aprender la operación elemental de dividir o para recordarla si sueles emplear la calculadora para realizarla.

Sólo tienes que introducir el dividendo y el divisor y comenzar a dividir de igual forma que lo harías sobre una hoja de papel.

Si te equivocas, tienes la opción de volver a repetir la operación desde el principio.

Sobre el programa

Hemos empleado técnicas de movimiento del cursor para imitar el desplazamiento de tu bolígrafo cuando realizas la operación sobre el papel.

No intentes engañar al ordenador con operaciones imposibles, estas situaciones se consideran en las líneas 160, 170 y 180

Las líneas comprendidas entre la 270 y 460 comprueban el resultado de las operaciones parciales a medida que vas realizando la división.

```
10 REM=======================
20 REM  APRENDIENDO A DIVIDIR
30 REM=======================
40 DIM A$(20),B$(20)
50 CLS
60 PRINT:PRINT:PRINTTAB(6)"INTRODUCE EL DIVIDENDO:":INPUT DEN$
70 PRINTTAB(6)"INTRODUCE EL DIVISOR:":INPUT NUM$
80 CLS
90  REM ========================
100 REM DEFINICION DE VARIABLES
110 REM Y DIBUJO DE LA OPERACION
120 REM ========================
130 LET X$=""
140 LET DEN=VAL(DEN$)
150 LET NUM=VAL(NUM$)
160 IF DEN<0 OR NUM<0 THEN CLS:PRINT:PRINT:PRINTTAB(5) "HAS INTRODUCIDO ALGUN NU
MERO NEGATIVO":FOR J=1 TO 1600:NEXT J:GOTO 50
170 IF NUM=0 THEN CLS:PRINT:PRINT:PRINT:PRINTTAB(5)"NO SE PUEDE DIVIDIR POR CERO
":FOR J=1 TO 1600:NEXT J:GOTO 50
180 IF DEN-NUM<0 THEN CLS:PRINT:PRINT:PRINTTAB(5) "EL DIVIDENDO HA DE SER MAYOR"
," QUE EL DIVISOR":FOR J=1 TO 1600:NEXT J:GOTO 50
190 LET LNUM = LEN (NUM$)
200 LET LDEN = LEN (DEN$)
210 LOCATE 20-LDEN,5:PRINT DEN$
220 LOCATE 24,5:PRINT NUM$
230 LOCATE 21,6:PRINT "L__________"
240 LET COCI=FIX(DEN/NUM)
250 LET RES=DEN-(COCI*NUM)
260 LET CUEN=LNUM:LET DES=0:LET RA=0
270 REM =======================
280 REM  INICIO DE LA OPERACION
290 REM =======================
300 LET X$=LEFT$(DEN$,LNUM)
```

```
310 IF (VAL(X$)/NUM)<1 GOTO 440
320 LOCATE 22+RA,7:A$(RA)=INPUT$(1):PRINT A$(RA)
330 LET LO=LEN(X$):LET DES=DES+1:LET RA=RA+1
340 LET X$=""
350 FOR I= 0 TO (LO-1)
360 LOCATE 20-LDEN+CUEN-I-1,5+DES:B$(I)=INPUT$(1):PRINT B$(I)
370 LET X$=B$(I)+X$
380 NEXT I
390 FOR P=LO TO 1 STEP-1
400 LET D$=LEFT$(X$,P)
410 IF VAL(D$)=0 THEN LET X$=RIGHT$(X$,LO-P)
420 NEXT P
430 GOTO 470
440 IF DES=0 THEN LET X$=LEFT$(DEN$,LNUM+1):LET CUEN=CUEN+1:GOTO 310
450 IF DES<>0 THEN LOCATE 22+RA,7:A$(RA)=INPUT$(1):PRINT A$(RA):LET RA=RA+1
460 LET H$=X$
470 REM ===============
480 REM BAJAR UN NUMERO
490 REM ===============
500 IF CUEN=LDEN THEN 580
510 LET X$=X$+MID$(DEN$,CUEN+1,1)
520 LOCATE 20-LDEN+CUEN,5+DES:PRINT MID$(DEN$,CUEN+1,1)
530 LET CUEN=CUEN+1
540 GOTO 310
550 REM=========================
560 REM COMPROBACION DEL RESULTADO
570 REM=========================
580 LET D$="":FOR H=0 TO RA-1
590 LET D$=D$+A$(H)
600 NEXT H
610 LET A$="N"
620 IF VAL(D$)=COCI AND (VAL(X$)=RES) THEN PRINT:PRINT:PRINT:PRINT:PRINTTAB(5)"E
S CORRECTA"
630 IF VAL(D$)<>COCI OR VAL(X$)<>RES ORVAL(H$)<>RES THEN PRINT:PRINT:PRINT:PRINT
TAB(5)"NO ES CORRECTA":PRINT:PRINTTAB(3)"QUIERES INTENTARLO DE NUEVO(S/N)":A$=IN
PUT$(1)
640 IF A$="S" THEN GOTO 80
650 FOR T=1 TO 1000:NEXT T
660 END
```

Modificaciones para otros equipos

El programa es válido para el ordenador MSX. Si tienes un IBM sólo tienes que invertir el orden de los argumentos en las sentencias locate según el siguiente cuadro.

IBM

210 locate 5,20-lden
220 locate 5,24
230 locate 6,21
320 locate 5+des,20-lden+cuen-i-1
450 locate 7,22+ra
520 locate 5+des,20-lden+cuen

COMMODORE

Sustituye la sentencia cls por print chr$(147) en las lineas 50,80,160,170,180
450 sustituye a$(ra)=input$(1) por input ;a$(ra)
630 sustituye a$=input$(1) por input ;a$

AMSTRAD

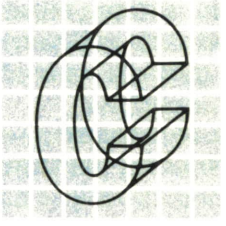
450 sustituye a$(ra)=input$(1) por input ;a$(ra)
630 sustituye a$=input$(1) por input ;a$

SPECTRUM

Utiliza variables alfanuméricas de una sola letra en las líneas 300 y 510.

```
300 let x$=den$(1 to lnum)
310 then goto 440
320 input;a$(ra) :print at 7,22+ra ; a$(ra)
360 input b$(i) :print at 5+des,20-lden+cuen-i-1;b$(i)
400 let d$=x$(1 to p)
405 let lx=len(x$)
410 if val(d$)=0 then let x$=x$(lo-p to lx)
440 if des=0 then let x$=x$(1 to lnum+1):let cuen=cuen+1:goto 310
450 if des<>0 then input a$(ra):print at 7,22+ra;a$(ra):let ra=ra+1
510 let x$=x$+den$(cuen+1 to cuen+1)
520 print at 5+des,20-lden+cuen;den$(cuen+1 to cuen+1)
630 sustituye a$=input$(1) por input a$
660 GOTO 9999
```

La máquina analítica de Babbage



COMENCEMOS hablando someramente sobre su creador.

Babbage fue todo un personaje. Su biografía es interesantísima, sorprendente. Su actividad recuerda a los grandes hombres del Renacimiento italiano..., aunque con un componente inglés, desde luego. Como Leonardo Da Vinci, se interesó por las matemáticas y por las máquinas de todo tipo. Aunque fue tan utópico como él, sus trabajos tenían siempre una aplicación inmediata, a veces banal, a primera vista. Como buen inglés, no era tan artista y no intentó pintar ni *Giocondas* ni *Cenas,* ni otras sutilezas para recreo de la vista. Sin embargo, se preocupó por mejorar el servicio de Correos, llegando a conclusiones revolucionarias, que simplificaban enormemente las tareas. Sus teorías serían aplicadas unos años más tarde, porque evidentemente, en el momento de ser emitidas habían causado molestias y desconfianza.

Babbage fue a la vez amable e irritable, social y solitario. Fue un verdadero visionario, que se anticipó en sus conclusiones a todos sus contemporáneos.

Nuestro hombre nació en un precioso pueblo de Devonshire, Totness, muy cerca de la costa de Torquay donde actualmente estudian inglés durante el verano multitud de aspirantes al dominio de este idioma. Su padre era banquero, detalle fundamental para comprender tanto la educación como la dedicación de Charles a sus ideas. El hecho de poder disponer de su tiempo para dedicarlo a estudiar, exactamente lo que uno desea, es un factor importante. Unido a una inteligencia notable, un tesón, sin duda a prueba de desilusiones, y un sentido del ridículo distinto del español, le hizo alcanzar cotas de razonamiento muy altas, y una visión de futuro como pocos hombres han tenido.

Pasemos a estudiar su creación de juventud: la máquina de diferencias.

Las máquinas de calcular automáticas, del siglo XVII, no eran realmente automáticas. Cada operación necesitaba de un operador, que manipulase la máquina, y, además, era necesario llevar un control paralelo, sobre papel, para comprobar que se había realizado la secuencia tal y como estaba prevista. También, y en cada paso parcial, este operador debe ir leyendo las cifras que se van obteniendo, tomando las decisiones de cuáles son las siguientes operaciones a llevar a cabo. Evidentemente, esas máquinas no eran automáticas, aunque sí supusieran un enorme avance para el cálculo en su época. Mejorar la velocidad de cálculo tampoco suponía un hito a alcanzar por las máquinas. Aunque los cálculos se realizaran en muy poco tiempo, siempre habría que sumarle el de los controles y manipulaciones, muy superior, de forma que al mejorar la velocidad de cálculo, apenas si se reduciría el tiempo real. No era, por tanto, interesante investigar por ese camino.

Es fácil para un hombre actual imaginar máquinas que no necesiten algún tipo de control humano. No lo era ni siquiera hace cien años, cuando Julio Verne era un loco que imaginaba cosas inalcanzables. Y ese fue uno de los grandes méritos de Babbage: fue capaz de considerar factible la fabricación de máquinas automáticas de calcular para las que no fuera necesaria intervención humana desde el comienzo de los cálculos.

Como hemos dicho, de entre los personajes que se interesaron en proyectar máquinas de calcular, desde luego el que obtuvo mejores resultados fue Charles Babbage. Este investigador fue atípico totalmente: fue profesor en Cambridge, en el Colegio de San Lucas, ¡pero no dio clase! Evidentemente debía ser valioso para que se lo permitieran. Desde luego lo era. Investigó entre varios posibles sistemas docentes para impartir las clases de matemáticas haciéndolas más asequibles para los alumnos. Posiblemente estas investigaciones le valieron el ser relevado de darlas. En otra de sus facetas creó unas tablas actuariales de mortandad, que eran las únicas fiables de la época. Y como sabe el lector, eso no fue todo, se interesó sobre los sistemas tipográficos existentes, organizó la Oficina Postal, colaboró con la industria de alfileres, estudió algunas reformas en los ferrocarriles, conduciendo él mismo un tren que circulaba los domingos, y con el que efectuaba medidas dinamométricas. Todo un estudioso. ¿Cómo no iba también a colaborar en la mejora del funcionamiento de la sociedad real a cuyo control está sujeto el observatorio de Greenwhich?

Las modificaciones para el sistema de correos eran, como cabía suponer, correctas y útiles, pero no fueron aceptadas en el momento. Su teoría era que el coste del envío de una carta o paquete se veía influido en mayor medida por los costes de recogida, clasificación y expedición que por la distancia del envío. Por tanto, indicaba normas para sistematizar los pesos, formas, etc., y cobrar cánones unificados. Como puede verse, la idea es absolutamente práctica, y naturalmente en nuestros días no hay oficina de correos que no adopte este tipo de medidas, junto con otras más, naturalmente.

Respecto de las tablas de todo tipo (logarítmicas, mortandad, etc.), que tanto preocupaban a Charles, y que quizá fueran la razón primera que le llevó a construir su máquina, su idea era mejorarlas, eliminar sus muchísimos errores. Como vemos, no se conformó simplemente con manejar tablas logarítmicas, para su cátedra de matemáticas. También creó unas tablas de mortandad muy fiables, tanto que fueron utilizadas en Inglaterra como en parte de Europa. Para ello, se desplazó a la Compañía Equitativa de Seguros de Vida, de la que obtuvo el asesoramiento necesario para desarrollar su labor. Como puede verse, ningún tema le parecía banal, todo era interesante, mejorable, modernizable, en suma.

De la personalidad de Babbage parecían haberse olvidado todos durante un siglo. Los ordenadores electrónicos, «hijos», en cierto modo de sus ideas, han recuperado su biografía, interesante y a la vez divertida.

Sabemos, por ejemplo, que «a pesar de su inteligencia», a veces, se comportaba de forma estúpida y puntillosa (o tal vez sólo deseaba algo de publicidad). En el libro de Bowden, el autor explica cómo Babbage envió una nota al gran poeta inglés Tennyson con ocasión de la aparición de su poema «La visión del pecado». Tennyson cantaba en su lirismo como:

En cada instante muere un hombre
Y en cada instante nace otro...

Babbage le envió una nota indicándole la falta de veracidad del poema: aunque no se ajustara tampoco a la verdad, sí sería suficientemente exacto que el poema glosara:

En cada instante muere un hombre
Y en cada instante nace uno y un dieciseisavo...

¿Humor o insensatez? No lo sabemos. Sus manías eran muchas (entre otras, no soportaba a vendedores ni músicos ambulantes que vagaban por la ciudad, ni parecían importarle en absoluto sus penalidades económicas, ni sus dificultades para sobrevivir). También es cierto que tampoco tenía pelos en la lengua con los poderosos, llámense personas, o instituciones. Llegó a decir de la Royal Society (que lo había condecorado) que él había sido su salvador, «rescatándola del desprecio de su país y del ridículo de los demás países».

Babbage construyó durante su vida dos máquinas distintas: la máquina de diferencias y la máquina analítica. La primera de ellas tiene muchas menos pretensiones que la segunda. De hecho, llegando a un punto, Babbaje dejó de interesarse por ella, pasándose al estudio y diseño de la segunda, verdadera precursora de los primeros ordenadores. (Debemos decir que el término «construyó» no es del todo correcta, ya que diseñó y construyó en parte sus máquinas, pero no llegó a terminarlas.)

Estudiemos someramente la Máquina de Diferencias. Babbage comenzó a trabajar en ella a la edad de veinte años. Fue concebida para calcular e imprimir tablas de funciones matemáticas. (Como hemos dicho, las ta-

blas de logaritmos de la época estaban casi siempre llenas de errores y era un tema grave, ya que a veces se producían hasta catástrofes de navegación por cálculos equivocados.)

La máquina se llamaba «de diferencias» (Δ, Δ^2y, Δ^3y) porque calculaba las diferencias entre los sucesivos valores de una función. Veamos una tabla típica de diferencias de una determinada función, como, por ejemplo, $y = 7x^2 + 3x + 1$. Como se trata de una ecuación de segundo grado, las segundas diferencias deben ser constantes, y las terceras (y, naturalmente), las sucesivas deben ser cero. (Esto ocurre con todos los polinomios, en los que al calcular las diferencias, siempre existen a partir de cierto grado unas que son constantes, y las siguientes cero. Como cualquier función puede representarse con la exactitud que se desee, mediante polinomios, y los valores de cualquier polinomio mediante sumas, el problema ya estaba resuelto.)

La máquina debe llevar un número de registros conectados mediante algún mecanismo para permitir realizar adiciones de uno a otro. Introduciendo en los registros iniciales los valores correctos, se puede ir paso a paso obteniendo los sucesivos valores de la función. Todo ello sin más que ir realizando reiterativamente adiciones.

Veamos, por ejemplo, una tabla de diferencias de un polinomio de tercer grado:

El polinomio puede ser, por ejemplo, x3 - 5x + 4

La tabla de diferencias, puede ser la siguiente:

x	y	Δ	Δ^2y	Δ^3y
1	0	-4	6	6
2	2	2	12	6
3	16	14	18	6
4	48	32	24	6

Este sistema de diferencias también es excelente para comprobar tablas. En nuestro caso, debe comenzarse de izquierda a derecha. Los errores se detectan inmediatamente en el punto en el que las diferencias deben ser constantes, y pueda aparecer un valor que no sea el mismo.

La máquina primera de Babbage era relativamente pequeña, sólo tenía tres registros, y, por tanto, tabulaba sólo funciones de segundo grado, con ocho decimales. La máquina era muy interesante y funcionaba muy bien. El gobierno se interesó por ella y a través de la Royal Society (que administraba el observatorio de Greenwich), decidió apoyarle para construir otra que alcanzara mayores cotas de exactitud: polinomios de hasta sexto grado, y veinte decimales. La verdad es que la Royal Society tuvo enorme paciencia con Babbage, que fue poco a poco aburriéndose de un tema que ya había estudiado y pasó a interesarse por multitud de otros temas, dejando el proyecto morir poco a poco. Había seleccionado para la construcción de su máquina a Joseph Clement, que tenía un serio prestigio en la fabricación de mecanismos. Clement era un magnífico artesano, pero, como hemos indicado muchas veces, los materiales de la época no eran demasiado fiables. Finalmente, y tras veinte años de espera y unos gastos enormes (casi veinte mil libras de la época), Babbage perdió el apoyo oficial y sin la compensación económica que le incentivaba a seguir, abandonó el proyecto.

¿Sabía usted que...

Narraremos también al lector algunas «historietas» sobre otros inventos atribuidos a Babbage.

Uno de ellos, muy útil por cierto, aunque lo sería más en Holanda, consiste en un sistema de «recuento de las vacas», cuando éstas vuelven al establo. Siento no poder informar al lector con más exactitud sobre este invento, y prometo hacerlo más tarde, si encuentro la oportunidad.'

Otro, en apariencia algo más serio y práctico en nuestros días (aunque posiblemente en su tiempo, el recuento de vacas fuera tarea más frecuente que hoy), es un medidor de velocidades, muy práctico para los ferrocarriles, y que Babbage consideraba que debía llevar incorporado cada locomotora.

También se interesó mucho por los sistemas de seguridad de todo tipo, desde puertas (diseñando llaves antirrobo, etc.), hasta códigos para los faros de la costa. El sistema que se utiliza en nuestros días, de ocultar la luz del faro, siguiendo siempre la misma secuencia en tiempo, es otro de los trabajos de investigación de Babbage.

Entre los proyectos que llevó a cabo sobre seguridad, se encuentra el ofrecido a la Red de Ferrocarriles. Sabemos que Babbage conducía un tren los domingos (cuando el tráfico era mucho menor) y efectuaba medidas dinamométricas. También estudiaba las irregularidades del terreno y sus influencias en la locomotora.

No parece que a Babbage le hiciera mucho impacto recibir la medalla de oro de la Royal Astronomic Society. Sencillamente, ese apoyo le sirvió para seguir trabajando en proyectos, algunos de ellos complicados e interminables.

Hacia 1832, Babbage comenzó a perseguir otra idea, desde luego más revolucionaria de las que tuvo en su vida. Se trataba de otra máquina, a la que llamó Máquina Analítica, y de la que hablaremos a continuación.

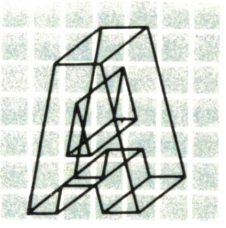
Respecto de la máquina de diferencias, cuya idea fue abandonada por Babbage, hubo otros que la siguieron, llegando a construirse máquinas con el mismo principio, que resultaron muy útiles para la elaboración de tablas de todo tipo. Entre los que tuvieron éxito en esta labor se encuentra Georg Scheutz, ingeniero sueco que construyó con la ayuda de su hijo varias máquinas que alcanzaban polinomios de hasta cuarto grado, con catorce decimales. El gobierno inglés, para quien la desesperanza es un vicio en el que no debe caer todo funcionario que se precie, compró (por fin) una de estas máquinas suecas de diferencias, para confeccionar unas tablas de mortandad, que salieron a la luz pública en 1864, cuarenta años más tarde de sus primeras inversiones en la máquina de Babbage.

Hoy día, siguen existiendo copias comercializadas de la Máquina de Diferencias. La Nattional Accounting Machine es ni más ni menos que una máquina como la de Charles Babbage, con seis registros de doce dígitos. En este caso, lleva también un sistema de impresión, y un pequeño teclado para introducir los valores con mayor facilidad.

De la máquina de Babbage, así como la de Scheutz existen algunos ejemplares que se conservan en el museo de Ciencias de Londres.

Babbage fue un verdadero «cerebro». Fue un matemático brillante, un ingeniero serio, y un hombre ingenioso. (Todo ello sin considerar su enorme aportación a la Informática, como creador de la máquina analítica.)

Gráficos por ordenador



A modo de introducción a este importante tema que iremos desarrollando, recordemos someramente cómo se fue gestando la idea de utilizar los ordenadores algo más que para resolver problemas, dando resultados concretos y muy aburridos.

Hablaremos en otros capítulos de los lenguajes adecuados, animación, etc. Muchos se siguen aún preguntando qué interés puede tener el diseño por ordenador. La pregunta puede contestarse retóricamente, es decir, ¿qué interés tiene la fotografía, por ejemplo?

Hemos visto que la fotografía es utilizada para innumerables tipos de trabajo, además de ser una afición muy agradable. La fotografía se utiliza en documentos, en peticiones de trabajo, en informes oficiales, ¡en casi cualquier actividad!

Y si se utiliza una fotografía, como explicación, por ejemplo, de cómo es un determinado mueble, que no tenemos en stock, pero del que sí podemos disponer, si lo desea el cliente, ¿por qué no utilizar un sistema de dibujo por ordenador, por ejemplo, para diseñar un mueble, absolutamente a gusto del cliente para que una vez contento con su propio diseño, nos lo encargue?

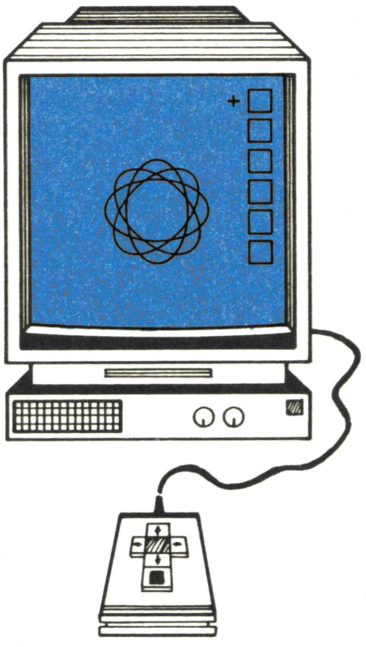
La utilidad del diseño por ordenador es evidente. La razón por la que no se ha utilizado demasiado en los años anteriores es muy clara, se trataba principalmente de un problema económico. Sin embargo, desde los años 50, el diseño por ordenador ha ido creciendo paulatinamente, hasta hacerse con un mercado amplio. Mucha culpa de ello la tiene la caída de los precios de ordenadores y componentes, que ha propiciado que casas de software a su vez vean los posible productos competitivos, y, por tanto, se lancen en este campo, de posibilidades infinitas.

Los ordenadores, desde sus comienzos han trazado curvas y diagramas, pero el hombre siempre ha tenido un enorme interés por representar lo que le rodea de una forma vívida y descriptiva (lo prueban los dibujos de animales, etc., de las cavernas). Los programadores de hace unos años dibujaban cuadros famosos, aprovechando las matizaciones más o menos claras de determinadas letras. Y así hemos visto todo tipo de cuadros famosos, desde Mona Lisa, hasta cristos de Dalí.

La utilidad de los diagramas para representar números, coyunturas, etc., es evidente. Probablemente, su principal utilidad sea su enorme rapidez. Pero ya hemos dicho que la creatividad humana no se conformaba con esos gráficos, claros y exactos. Era necesario pasar al diseño, al dibujo por ordenador.

En los primeros años, se crearon sistemas gráficos para organismos y compañías de economías pujantes como, por ejemplo, para algunos ejércitos (el sistema SAGE de defensa y control aéreo) o para compañías de aviación, simulaciones, etc., u otras compañías de automóviles de gran lujo (Rolls-Royce y General Motors). El «Sketchpad» creado por Ivan Sutherland, en 1963, tenía miras más altas. Era en cierto modo interactivo, ya que disponía de un menú de símbolos y de un lápiz electrónico.

En los años 60, los sistemas de dibujo de pantallas por ordenador tienen todavía precios altísimos. Los dibujos en la pantalla se creaban mediante un tubo de rayos catódicos y un rayo de electrones que dibujaba unas líneas (o vectores) que conformaban el dibujo sobre un recubrimiento de un determinado tipo de fósforo. El dibujo permanecía en la pantalla sólo una pequeña fracción de segundo antes de desvanecerse, por lo que era necesario dibujarlo continuamente, al menos 30 veces por segundo. (La razón de que durara tan poco en la pantalla se debía al tipo de fósforo utilizado en su fabricación, que no mantenía apenas la imagen trazada por el rayo de



barrido.) Si todo lo anterior era un problema serio, en lo que respecta a las copias duras, éstas sólo podían obtenerse fotografiando la pantalla, o bien mediante un plotter, elemento carísimo. Pero si se deseaban más copias del dibujo, el problema se complicaba enormemente, ya que la memoria que debía almacenar toda la información sobre el dibujo debía tener un tamaño considerable, cosa que no siempre era posible. (Algo más tarde, aparecieron otros tipos de fósforo que mantenían la imagen por más tiempo, resolviendo el problema de tener que redibujarla tantas veces cada segundo.)

Los tubos de rayos catódicos de Tectronix, aparecidos en los años 60, supusieron un abaratamiento considerable, y ayudaron a muchas empresas a interesarse en el campo del dibujo por ordenador.

Sólo hacia finales de la década de los 70, con la aparición y posterior desarrollo de la microelectrónica, el precio baja de manera fulgurante. Ello se debe principalmente a que las memorias (elemento fundamental en estos diseños) han caído de precio tanto que se producen dos fenómenos paralelos: por un lado, los ordenadores se han abaratado muchísimo, y son ya accesibles para multitud de aplicaciones, y, por otro, son mucho más potentes, tienen memorias mucho más grandes, lo que permite que ordenadores de precios reducidos puedan realizar diseños de cierta calidad.

La primera sociedad para la promoción de los gráficos por ordenador se creó en 1963 (Society for Information Display-Sociedad para la Presentación de la Información mediante Gráficos por Ordenador). Algo más tarde, y bajo los auspicios de ésta, se creó la SIGGRAPH que en nuestros días cuenta con más de 35.000 socios.

La utilidad de los gráficos por ordenador es enorme. Un refrán antiguo dice «una imagen vale más que mil palabras». Como suele suceder con los refranes, basados en la experiencia, es una aseveración válida. Pensemos en un gráfico que describa cualquier conjunto de situaciones. En pocos segundos nos daremos cuenta en qué punto nos encontramos y podremos tomar la decisión adecuada.

Los terminales pensados para dibujar en el ordenador tienen una pantalla que imaginaremos compuesta por multitud de puntos, organizados en filas y columnas. La memoria del ordenador está relacionada con el número de puntos de la pantalla, o resolución, ya que debe tener capacidad para almacenar, al menos dos valores por cada uno de estos puntos, o elementos individuales de la pantalla. Realmente, dos valores en nuestros días, con la facilidad de poder adquirir monitores, etc., en color, constituyen muy poca información, ya que sólo toman nota de si está encendido o apagado. Si consideramos los colores, es necesaria mucha mayor información. Como una pantalla de 512 × 512 es una pantalla aceptable, piénsese en que la memoria debe aceptar información (varios datos) de 262.144 puntos. Creo que habremos aclarado porqué ha crecido el interés por el dibujo por ordenador con la aparición del chip.

Hemos hablado someramente de algo del hardware implicado en el dibujo por ordenador. El software no ha tenido un crecimiento parejo con el hardware. Normalmente, el software de los primeros años era creado por los propios fabricantes de ordenadores para sus ordenadores y se trataba simplemente de algunas rutinas sencillas para dibujo de curvas, puntos, líneas, etc., que podían «componer» dibujos. El usuario debía aguzar el ingenio, y aprovechando el material, dibujar sus ideas, o crear su propio software. Afortunadamente, los tiempos han cambiado, y, hoy día, existe abundante software gráfico, creado por firmas independientes de las casas fabricantes y, por tanto, susceptible de ser utilizado en muchas gamas de ordenadores. La mayor parte de este software está escrito en Fortran, ya que se considera que es el lenguaje que com-

porta una mayor facilidad de cambio de ordenador.

También se han creado otros lenguajes, pensando específicamente en los problemas gráficos. Uno de ellos es el MIRA, que introduce un nuevo concepto, el del tipo abstracto. Este lenguaje, define los objetos por sus cualidades propias, pero no contempla en absoluto el sistema seguido en su formación. Así, define el objeto gráfico como un tipo, cuyas variables gráficas pueden manipularse y no considera en absoluto la forma o el método de su construcción. Este lenguaje es una extensión del lenguaje Pascal y se considera para dibujar un espacio de dos o tres dimensiones.

Para hacernos idea de cómo funciona este lenguaje, pensemos en un diseñador de muebles para cocinas que, aunque pueda resultar extraño, no tiene ningún diseño realizado, le ofrece la posibilidad de amueblar su cocina con un diseño creado por usted.

El lenguaje considera un mueble como un conjunto de parámetros, relacionados, que formarán cada mueble sólo en el momento de su creación. Así, su armario puede ser más o menos estrecho o alto (con tal de que sea un armario), con la puerta entera, con cajones, con una puerta y un cajón, etc.) que son los datos que el usuario va introduciendo, seleccionándolos de entre una gama grande. Una vez obtenido el mueble con puerta o cajones, alto o bajo o ancho o en ángulo recto, se le pondrán los tiradores...

Como puede verse, es difícil que se produzcan dos armarios de cocina iguales, de entre la enorme gama de posibilidades. El conjunto de parámetros elegidos son los que conformarán el objeto.

Pasemos ahora a ver qué campos trata el dibujo por ordenador. La investigación en este campo se ha dirigido principalmente a obtener imágenes de una calidad superior, por un lado, y, por otro, a mejorar los sistemas interactivos entre el ordenador y creador, incluidos los lenguajes gráficos.

Para que las imágenes resulten reales, con calidad, observemos qué «ingredientes» llevan implícitos:

— Deben tener sensación tridimensional. Nos movemos en un espacio de tres dimensiones.

— Debe considerarse su textura. No es igual la piel de una fresa, por ejemplo, que la de un pepino.

— También debe considerarse que los objetos no son, por lo general, transparentes, y, por lo tanto, tapan aquello que se encuentra detrás de ellos y también, naturalmente, ciertas partes de ellos mismos.

— La iluminación es fundamental. Dentro de esta iluminación, existen dos tipos distintos: uno de ellos, la iluminación uniforme y difusa, no ofrece interés alguno, ya que sólo afectaría a la brillantez general de los colores. Los focos de luz, por el contrario, tienen mucho más interés, ya que pruducen sombras distintas sobre el objeto y objetos iluminados, favoreciendo la sensación de tridimensionalidad de la imagen. Además, aquí también aparecen las sombras causadas por cada uno de los objetos sobre los demás que componen la imagen.

Las transparencias, son difíciles de obtener. Turner Whitted ha conseguido efectos de una calidad irreprochable.

Veamos a continuación otros factores que mejoran la calidad de la imagen. La resolución es fundamental. Evidentemente, si se pueden controlar individualmente los parámetros de muchos más puntos, la imagen dará una sensación de realidad mayor.

En los ordenadores personales, la producción de imágenes de mucha calidad presenta muchos problemas. Por un lado, la memoria de que disponen, aunque grande en comparación con la de los ordenadores de los años 60 y 70, es a veces insuficiente. Por otro lado, la resolución suele ir vinculada al número de colores disponibles simultáneamente, de forma que a mayor número de colores disponibles, menor resolución. Para que una imagen cualquiera dé sensación de realidad, la resolución debe ser al menos de 512 × 512 pixels. En lo que se refiere a los colores, la paleta debería contener al menos 256, para poder obtener gamas en degradé, que puedan representar sombreados y transparencias con un mínimo de credibilidad.

En los ordenadores pequeños, en los que con una resolución no muy buena se puede disponer de, por ejemplo, ocho colores, una de las posibles soluciones, es crear gamas de colores distintos, mezclando dos o más colores pixel a pixel en una determinada superficie, de forma que el efecto óptico producido sea un color distinto de la mezcla de los dos colores que lo han formado.

TERMINOLOGIA

Algoritmo. Término muy utilizado en informática. Es un conjunto de operaciones matemáticas cuyo fin es resolver un determinado problema. El diseño de gráficos utiliza los algoritmos para resolver problemas de todo tipo, como, por ejemplo, la eliminación de las líneas que aparecen ocultas, manipulación del color, etc.

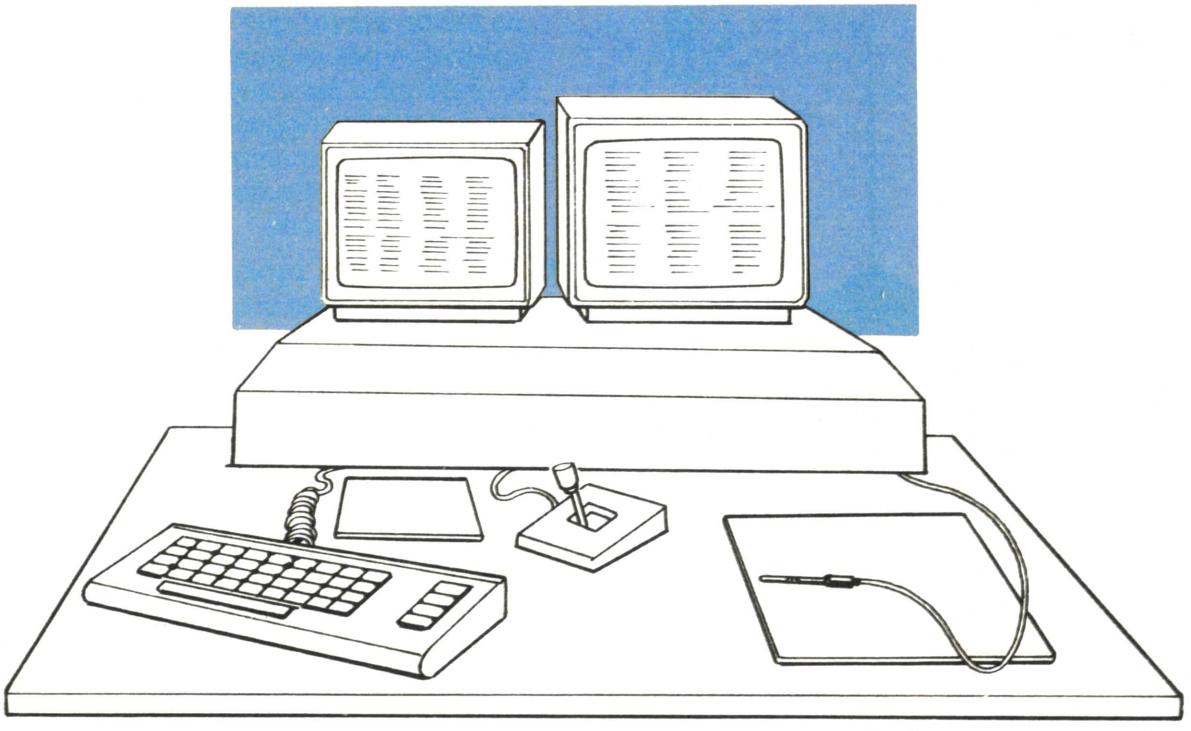
Animación. Acción por la cual se dota de «movimiento» artificialmente a un dibujo. Estas técnicas utilizan una secuencia de imágenes muy semejantes, pero que van cambiando poco a poco y que se van proyectando a una velocidad tal que el ojo humano no puede observar las discontinuidades. Las velocidades más utilizadas son 24 ó 25 imágenes por segundo, o 18, según la calidad necesaria (en EE.UU. se utilizan 30, debido a que la electricidad que genera a 60 hercios, frente a Europa, que son 50), y escogiendo ese valor se reducían los problemas de transmisión. Cuando se realizan intercambios entre EE.UU. y Europa, existen unos dispositivos que solventan el problema, bien repitiendo ciertas imágenes, bien aumentando la velocidad.

Bezier, curvas y superficies. Bezier desarrolló unas técnicas matemáticas para dibujar curvas y superficies que debían cumplir determinadas condiciones: tenían unos puntos de comienzo y final fijos y, además, se veían influenciadas por otros puntos intermedios. Bezier desarrolló estas técnicas cuando trabajaba en el diseño de coches para la fábrica de automóviles Renault.

Los puntos intermedios no tienen influencia alguna en la curva en ninguno de los dos puntos de comienzo o de final. En el resto de los puntos de la trayectoria, los puntos intermedios tienen una influencia de tipo atracción gravitacional. Este tipo de curvas se utilizan mucho en animación. (Si lo explicamos sencillamente, colocando una serie de puntos sobre un plano, por ejemplo, si nos moviéramos desde el primero al último, iríamos sufriendo progresivas influencias de los más próximos, y se irían atenuando las influencias de los que iban alejándose.)

Brillo. El ojo humano no aprecia la intensidad de la luz por la brillantez de los objetos que ve. Para tomar este dato, habría que considerar la orientación de la superficie. Sin embargo, cuando se va a realizar un dibujo por ordenador, se suele considerar que las superficies reflejan la luz con la misma intensidad en todas las direcciones. También suele considerarse que el foco de luz se encuentra a una distancia tal que los distintos recorridos desde el foco a las diferentes partes del objeto son insignificantes. Para calcular el brillo de un determinado punto de una superficie, es necesario saber el ángulo de incidencia, y la intensidad de la fuente luminosa. Si la fuente luminosa tiene una intensidad de uno, los niveles de brillo variarán de cero a uno, dependiendo del ángulo de incidencia, que es el que se utiliza para organizar los niveles de color.

Business graphics (gráficos empresariales). Gráficos pensados principalmente para ayudar en la toma de decisiones de la



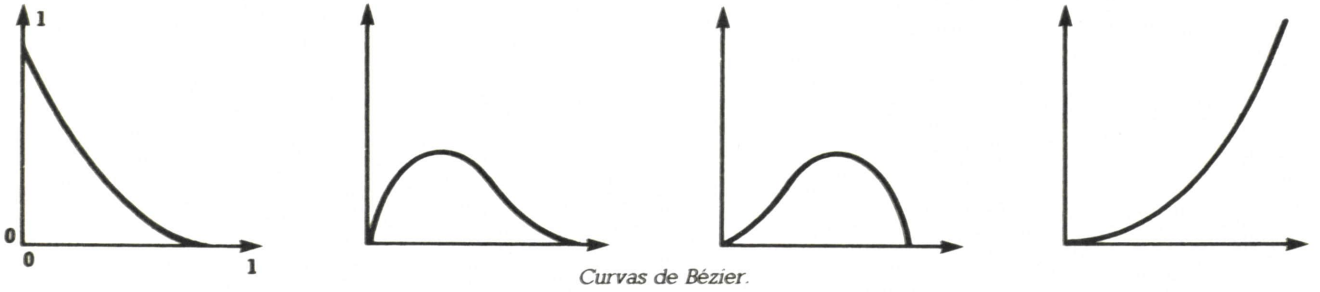
Sistema CRD.

empresa. Constituyen una interpretación condensada de los datos obtenidos. Existen muchos tipos útiles para estas tareas: histogramas, gráficos circulares (o de queso), etc. Hoy día, muchos periféricos pueden realizar diagramas fácilmente tomando los datos adecuadamente de los ficheros no gráficos.

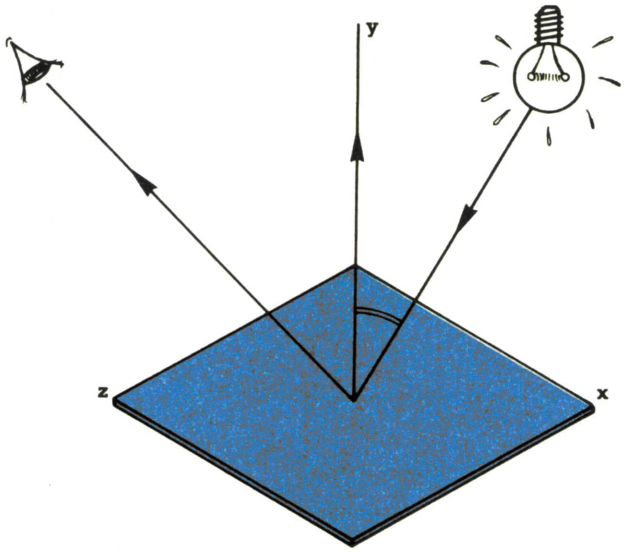
CAD. Computer Aided Design (Sistema de Diseño ayudado por Ordenador). Es un sistema que permite al usuario diseñar, utilizando un ordenador de forma interactiva. El usuario dispone de un conjunto de comandos muy sencillos para trabajar en su diseño.

Un sistema CAD lleva un terminal gráfico, un sistema de presentación visual, y una tableta. El usuario puede enviar ciertos comandos al ordenador, y también puede utilizar un lápiz electrónico para comunicarse con el ordenador. Este le envía sus respuestas mediante imágenes y mensajes. Estos sistemas se utilizan en multitud de aplicaciones, como diseño de circuitos integrados, construcción de barcos, diseño textil, e ingeniería en general. Existen muchos sistemas CAD que llevan incorporados procedimientos de fabricación (CAM). Los más sofisticados, disponen de caracerísticas de cálculo como volúmenes, momentos de inercia, centros de gravedad, que mejorar enormemente el diseño, ya que resuelven problemas como el tipo de grúas que moverán el material diseñado, la mejor forma para apilar elementos, como tazas, etc.

Coordenadas cartesianas. Sistema de ejes mediante el cual, especificando dos valores (uno referido al eje x y otro al eje y) se pue-



Curvas de Bézier.



de especificar una posición única de cada uno de los puntos de la superficie. Estas coordenadas se cortan con un ángulo de 90 grados. Los sistemas de coordenadas simplifican enormemente el problema de almacenar en la memoria datos sobre objetos, y el de digitalizar imágenes, ya que una vez reducida la imagen a un conjunto de valores, éstos se pueden manipular perfectamente por el ordenador. Por ejemplo, para desplazar una imagen en sentido horizontal, se pueden incrementar o decrementar sus coordenadas x. Para aumentar el tamaño de la imagen, se puede «escalar», es decir, multiplicar sus coordenadas por un determinado factor, mayor que uno, o realizar la multiplicación por un quebrado para disminuirlo. Para girar las imágenes se pueden utilizar funciones trigonométricas. En general, simplifican la manipulación de formas y constituyen puntos de referencia muy útiles para los diagramas.

Color. Sensación producida en la retina del ojo por una radiación electromagnética de una determinada longitud de onda. Las distintas longitudes de onda van produciendo las sensaciones de los distintos colores. Cuando decimos que un objeto es rojo, lo que ocurre es que la longitud de onda dominante es la correspondiente al rojo. Generalmente, no existen radiaciones de colores puros, sino un espectro en el que la longitud de onda dominante corresponde a ese color.

En la creación de dibujos por ordenador, la información referente al color se almacena mediante tres números que representan los tres colores primarios. No es, sin embargo, el mejor sistema de almacenar los valores del color.

Cromatismo. El color de un determinado objeto está formado por un espectro de ondas de distintas longitudes de onda, siendo algunas más intensas que otras. El cromatismo es un parámetro que mide la longitud de onda dominante, es decir, en cierto modo indica la «pureza» de un determinado color.

Hardware check. Sinónimo de comprobación automática.

Heurístico. Método de búsqueda de caminos para la resolución de problemas en el cual se obtienen las soluciones mediante evaluación de los progresos realizados hacia la obtención del resultado final. En contraposición a algorítmico.

Hex. (Véase **Hexadecimal**.)

Hexadecimal. Sistema de numeración en base 16. Los dígitos utilizados para representar los números son 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. A estos dígitos les corresponden los valores de 0 a 15. En cierto modo supone una abreviación del código binario, en el sentido de que es una combinación de valores binarios, pero los números resultan mucho más cortos.

High speed, arrastre. En adiciones paralelo, cualquier técnica utilizada para acelerar la propagación. En contraposición a "en cascada".

Hoja electrónica. Programa de uso general que presenta en la pantalla una tabla en la que se introducen fórmulas y datos fácilmente. Nada más introducir los datos (o al final de una sesión de adiciones de datos), la tabla se suele recalcular automáticamente.

Hollerith, código. Código binario de doce niveles, muy utilizado en la época de las tarjetas perforadas.

Home. Punto de partida, posición que toma el cursor en el extremo superior izquierdo de la pantalla.

Home computer (ordenador casero). Ordenador pequeño en dimensiones, de potencia pequeña o media, que se puede utilizar en actividades personales.

Horizontal, carácter de tabulación. Carácter que causa que la impresión se desplace hacia adelante a la próxima posición de tabulación, dentro de la misma línea de impresión o de presentación.

Hybrid circuit. Circuito integrado con la peculiaridad de que en el componente base del soporte se integran componentes discretos.

Hybrid, computer. Ordenador que utiliza tanto representaciones analógicas como digitales.

Hysteresis loop. (Véase **Magnética, bucle de histéresis**.)

I/O (Input/Output). Abreviatura inglesa de Entrada/Salida.

Identificador. Símbolo que sirve para identificar, indicar o nombrar un conjunto de datos.

Identity unit. Circuito lógico de entrada múltiple que sólo envía una señal de salida cuando las n señales de entrada son iguales.

Iddle time. (Véase **Tiempo de espera**.)

IEEE 488. Norma para conectar varios dispositivos a través de un BUS de 24 hilos, en paralelo. Permite que cada uno pueda comunicarse (enviar mensajes) con los demás (que reciben). Se utiliza en determinados periféricos (plotters) y en instrumentación.

Igual a. Operador de relación que suele indicarse con el signo =. También se utiliza en instrucciones de asignación.

Ilegal, carácter. Carácter o combinación de bits que no es válido según un determinado criterio o código. Por ejemplo, respecto a un alfabeto específico, los caracteres que no pertenecen a él.

Imagen. Representación matricial de los datos grabados en un soporte físico.

Imagen, disector de. En reconocimiento óptico de caracteres, transductor mecánico o electrónico que va detectando los distintos niveles de luz en determinadas zonas de un espacio muestra iluminado.

Impar. Lo contrario a par. En inglés «odd».

Impreso, formulario. Hoja donde se escriben los resultados (variables) de un proceso de datos (mediante la correspondiente impresora), pero donde están impresos previamente los rótulos, cabeceras, líneas divisorias, etc.

Impresora. Unidad periférica de salida de un ordenador que permite escribir textos sobre papel. Las características más importantes de las distintas impresoras suelen ser su velocidad de impresión (dada en caracteres por segundo, o líneas por minuto), y la calidad de letra (NLQ Near Letter Quality —calidad como de carta—), que es la mejor calidad de impresión. También es importante considerar el número de columnas (o ancho del papel), la técnica de impresión utilizada (matriz de puntos, macillos, cadena, etcétera), el tipo de papel que admiten, si pueden escribir caracteres gráficos o semigráficos, en color, etc.

Impresora paralelo. Impresora conectada al ordenador a través de un interfaz paralelo. Suelen tener este tipo de interfaz las impresoras de líneas (que escriben una línea completa cada vez).

Impresora serie. En contraposición a impresora paralelo. Impresora conectada al ordenador a través de un interfaz serie (un solo canal de datos por donde se transmiten los caracteres, uno tras otro). Las impresoras serie suelen ser del tipo "carácter a carácter", aunque hay impresoras de este tipo que trabajan conectadas a un interfaz paralelo.